21世纪高等学校规划教材 | 计算机应用



“十二五”普通高等教育本科国家级规划教材





Web技术导论
（第3版）

郝兴伟 主编

清华大学出版社

21 世纪高等学校规划教材 · 计算机应用

Web 技术导论(第 3 版)

郝兴伟　主编

清华大学出版社
北　京

内 容 简 介

本书首先介绍 Internet 和 WWW 中的主要概念、相关核心技术及 Web 的发展趋势；然后以 Web 中的 B/S 三层结构为主线，以具体的研发项目为背景，系统介绍 Web 应用系统开发中的相关问题，包括 Web 运行环境、HTML 和 XML、页面设计与制作、客户端编程和服务端编程问题，并提供了大量的案例和代码。

本书内容新颖，知识全面。本书可以作为高等学校计算机应用、信息管理及电子商务等专业的 Web 技术、Web 程序设计、互联网与 Web 编程等课程的教材，也可以作为互联网时代高等学校开设通识类课程的教材。

图书在版编目(CIP)数据

Web 技术导论/郝兴伟主编. --3 版. --北京：清华大学出版社，2012.8
(21 世纪高等学校规划教材·计算机应用)
ISBN 978-7-302-29629-4

Ⅰ. ①W…　Ⅱ. ①郝…　Ⅲ. ①网页制作工具－程序设计－高等学校－教材　Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字(2012)第 184196 号

责任编辑： 付弘宇　赵晓宁
封面设计：
责任校对： 焦丽丽
责任印制：

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn，http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载： http://www.tup.com.cn，010-62795954
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 185mm×260mm　**印　张：** 26.5　**字　　数：** 645 千字
版　　次： 2005 年 1 月第 1 版　2012 年 8 月第 3 版　**印　　次：** 2012 年 8 月第1 次印刷
印　　数： 1～　000
定　　价： .00 元

产品编号：049009-01

编审委员会成员

（按地区排序）

出版说明

随着我国改革开放的进一步深化，高等教育也得到了快速发展，各地高校紧密结合地方经济建设发展需要，科学运用市场调节机制，加大了使用信息科学等现代科学技术提升、改造传统学科专业的投入力度，通过教育改革合理调整和配置了教育资源，优化了传统学科专业，积极为地方经济建设输送人才，为我国经济社会的快速、健康和可持续发展以及高等教育自身的改革发展做出了巨大贡献。但是，高等教育质量还需要进一步提高以适应经济社会发展的需要，不少高校的专业设置和结构不尽合理，教师队伍整体素质亟待提高，人才培养模式、教学内容和方法需要进一步转变，学生的实践能力和创新精神亟待加强。

教育部一直十分重视高等教育质量工作。2007 年 1 月，教育部下发了《关于实施高等学校本科教学质量与教学改革工程的意见》，计划实施“高等学校本科教学质量与教学改革工程(简称‘质量工程’)”，通过专业结构调整、课程教材建设、实践教学改革、教学团队建设等多项内容，进一步深化高等学校教学改革，提高人才培养的能力和水平，更好地满足经济社会发展对高素质人才的需要。在贯彻和落实教育部“质量工程”的过程中，各地高校发挥师资力量强、办学经验丰富、教学资源充裕等优势，对其特色专业及特色课程(群)加以规划、整理和总结，更新教学内容、改革课程体系，建设了一大批内容新、体系新、方法新、手段新的特色课程。在此基础上，经教育部相关教学指导委员会专家的指导和建议，清华大学出版社在多个领域精选各高校的特色课程，分别规划出版系列教材，以配合“质量工程”的实施，满足各高校教学质量和教学改革的需要。

为了深入贯彻落实教育部《关于加强高等学校本科教学工作，提高教学质量的若干意见》精神，紧密配合教育部已经启动的“高等学校教学质量与教学改革工程精品课程建设工作”，在有关专家、教授的倡议和有关部门的大力支持下，我们组织并成立了“清华大学出版社教材编审委员会”(以下简称“编委会”)，旨在配合教育部制定精品课程教材的出版规划，讨论并实施精品课程教材的编写与出版工作。“编委会”成员皆来自全国各类高等学校教学与科研第一线的骨干教师，其中许多教师为各校相关院、系主管教学的院长或系主任。

按照教育部的要求，“编委会”一致认为，精品课程的建设工作从开始就要坚持高标准、严要求，处于一个比较高的起点上；精品课程教材应该能够反映各高校教学改革与课程建设的需要，要有特色风格、有创新性(新体系、新内容、新手段、新思路，教材的内容体系有较高的科学创新、技术创新和理念创新的含量)、先进性(对原有的学科体系有实质性的改革和发展，顺应并符合 21 世纪教学发展的规律，代表并引领课程发展的趋势和方向)、示范性(教材所体现的课程体系具有较广泛的辐射性和示范性)和一定的前瞻性。教材由个人申报或各校推荐(通过所在高校的“编委会”成员推荐)，经“编委会”认真评审，最后由清华大学出版

社审定出版。

目前，针对计算机类和电子信息类相关专业成立了两个“编委会”，即“清华大学出版社计算机教材编审委员会”和“清华大学出版社电子信息教材编审委员会”。推出的特色精品教材包括：

(1) 21世纪高等学校规划教材·计算机应用——高等学校各类专业，特别是非计算机专业的计算机应用类教材。

(2) 21世纪高等学校规划教材·计算机科学与技术——高等学校计算机相关专业的教材。

(3) 21世纪高等学校规划教材·电子信息——高等学校电子信息相关专业的教材。

(4) 21世纪高等学校规划教材·软件工程——高等学校软件工程相关专业的教材。

(5) 21世纪高等学校规划教材·信息管理与信息系统。

(6) 21世纪高等学校规划教材·财经管理与应用。

(7) 21世纪高等学校规划教材·电子商务。

(8) 21世纪高等学校规划教材·物联网。

清华大学出版社经过三十多年的努力，在教材尤其是计算机和电子信息类专业教材出版方面树立了权威品牌，为我国的高等教育事业做出了重要贡献。清华版教材形成了技术准确、内容严谨的独特风格，这种风格将延续并反映在特色精品教材的建设中。

清华大学出版社教材编审委员会
联系人：魏江江
E-mail：weijj@tup.tsinghua.edu.cn



第3版前言

进入20世纪90年代以后，随着互联网应用的日益普及，Web已经融入到人们生活的方方面面，Web技术也正在悄悄地影响和改变着人们的工作、生活和思维方式。在软件开发和计算机应用上，基于Web的计算机应用模式正悄然兴起，B/S结构在市场竞争中已经脱颖而出，C/S结构的系统风光不再。现在，无论是企业的电子商务，还是政府办公系统，可以说，80%以上的计算机应用都是基于Web的，使互联网这个现代的信息平台找到了最广阔的应用市场。今天，Web已经成为人们一种基本的技术素养，成为了一种文化现象。

身处教育和IT两个领域，一直对计算机软件技术的发展非常关注，希望所教授的知识和社会的需求紧密相关，培养的学生既有扎实、宽厚、系统的基础理论知识，又有优秀的开发能力。理论和实践的结合需要一个载体，对计算机软件来说，项目是再好不过的载体了。在软件项目的研发中，我们可以对相关的理论知识进行应用验证；反过来，在软件开发和应用中遇到的问题、难题，又推动了我们对理论问题的研究。

科研和教学就是这样的相辅相成，项目作为一种结合点，推动着科研和教学水平的不断提高。从2001年开始，我们开始了基于B/S三层架构的软件项目的研发工作，开发领域主要是E-learning与知识管理、虚拟实验室，还有一些通用的管理系统，例如会议管理、论文管理、用户服务支持系统等。在这些Web应用的开发中，遇到了许许多多的问题，也积累了很多的经验，有很多的体会。笔者不断地总结着研究和开发中的体会，在通过大学讲台和学生们交流、分享的同时，于2005年探索性地出版了《Web技术导论》一书，其目的就是要在高校中开设一门全面反映Web技术及应用的新的课程，书中的例题和案例大都来源于这些研发项目。2006年该书入选普通高等教育“十一五”国家级规划教材。此后经过三年多的积累和写作，2009年11月出版了《Web技术导论(第2版)》。

光阴荏苒，一晃又是三年过去了，在这三年中，笔者从未放下Web系统的研发工作。从2010年暑期开始直到2011年暑期，笔者利用整整一年的时间重写了“面向过程管理的网络教学平台”的几乎全部代码。从2011年底开始，一个新的Web系统平台，即“高等学校本科教学基本状态数据统计平台”项目已经启动，这是一项新的挑战，也充满着诱惑和快乐。在这些项目的研发中，笔者对Web系统的许多问题有了新的认识和体会。2011年底，“Web技术导论”课程被选为山东大学核心通识课程，同时，该课程也被教育部全国高校教师网培中心选为高校师资培训课程。笔者想在讲课以前，把自己三年来的知识积累写进教材中，这就是写作《Web技术导论(第3版)》的初衷。

本书保留了第2版的结构，几年来的教学实践表明了这种结构的科学性和合理性。但和第2版相比，本书在每一章的内容组织和结构上做了许多调整，例如，第1章增加了HTTP的讲解；第2章增加了虚拟主机和虚拟目录的知识；第3章对HTML进行了进一步凝练，加强了XML的讲解；第5章增加了正则表达式对象的介绍；第6章增加了SQL一节。对于第2版中许多陈旧的内容、没有实际应用的代码进行了删减。

本书在例题、习题上进行了更加精细的设计,全书共收集和讲解了150多段非常有用的程序代码,内容涉及HTML中的页面布局,CSS和图层的设计和应用,弹出式菜单、树形菜单的设计和不同的实现方法,表单数据的获取,用户输入的有效性验证,数据之间的类型转换,页面之间的参数传递,页面安全,服务端的文件和文件夹操作,数据库的访问和操作,SQL查询,AJAX异步通信等各种实用技术。

本书仍分为6章,主要内容如下。

第1章 Web基础。介绍Internet的产生和发展、万维网的概念、HTTP通信原理以及Web应用的概念;介绍Web相关的核心技术,包括Java技术、XML技术、Web服务等;然后介绍计算机应用模式C/S架构到B/S架构的演变,以及软件体系架构和开发模式的发展,特别是SOA架构及相关概念;最后对语义Web等新进展进行概要性介绍。

第2章 Web服务器的架设和管理。介绍Web服务器的概念、Web服务器的功能。主要讲解Windows Server 2003中IIS的配置和管理,讲解Apache和Tomcat的功能以及它们的关系、Apache和Tomcat的架设和管理,讲解虚拟主机和虚拟目录的概念及其配置方法。

第3章 HTML与XML基础。介绍标记语言的概念;然后详细介绍HTML的语法,对CSS技术、图层进行深入讲解,并安排大量的例子解释每个标记的含义和使用;然后讲解XML的本质特征,剖析了XML和HTML的本质区别;介绍XML开发工具XML Spy的应用。

第4章 网页设计与制作。网页作为Web应用的主要用户界面,在HTML基础上,加强了网页设计的讲解,包括页面功能与内容设计、页面布局设计、页面视觉设计以及页面效果设计等;然后,介绍可视化制作工具FrontPage的使用。

第5章 客户端编程。首先讲解Web浏览器的基本工作原理;然后讲解客户端脚本程序设计语言JavaScript、浏览器对象模型(BOM)、HTML文档对象模型(DOM)等;并讲解AJAX技术;通过四个综合案例,详细讲解JavaScript中菜单的实现、表单数据的有效性验证、表单数据的处理等问题,这些综合案例包含许多Web开发中所需要的代码。

第6章 服务端编程。首先介绍B/S三层结构的概念;然后简单讲解Java程序设计语言基础,介绍Java技术的特点、类与对象、接口、包等基本概念,介绍JavaBeans、Servlet服务器程序的概念,这些概念是开展基于Java技术的服务端编程的基础。本章的重点是JSP技术和数据库编程。在JSP技术中,讲解JSP的语法、JSP中的数据类型及其转换、JSP内置对象、JSP中的参数传递方法等内容;然后讲解服务端的数据库编程以及SQL,并给出两个综合性例子;讲解基于AJAX技术的在线聊天Web应用的整个开发过程;最后对常用的Java开发工具进行介绍。

作为互联网的用户和Web技术的开发者和实践者,同时作为一个公司派的高校教师,笔者希望这本书的知识结构和内容对于您了解Internet和WWW、学习Web开发、进行Web编程以及提高Web的应用水平等能有所帮助。也希望其中的大量例子在未来的Web研发中,对编程有所启发,节省宝贵的时间。软件开发是一个积累的过程,让我们一起在这种积累中进步,来享受成功的乐趣。

在本书写作的过程中,非常感谢我的同事巩裕伟、焦文江、杨兴强、韩振、阚铮和李蕴等多位老师的工作和提出的良好建议,还要感谢我的学生苏雪、常跃峰、崔旭、朱岩、田容雨、尤



凤英、董颖、张会昌、卢艳萍、田韶存等，他(她)们都参与了我们许多项目的研发工作，编写了大量的程序代码，祝愿他(她)们在以后的工作和生活中一切顺利，祝愿他们取得更大的成绩。还要感谢山东大学本科生院、山东大学研究生院的立项支持，感谢教育部全国高校教师网培中心对本书的厚爱，感谢清华大学出版社付弘宇编辑长期以来对本书的辛勤付出。

由于本书涉及的内容非常广泛，在深度和广度上很难做到完美，同时，也由于笔者的知识面和认识有限，书中肯定存在错误和不足，敬请各位同行和读者批评指正。

笔者的 E-mail：hxw@sdu.edu.cn，课程网址：http://gsl.sdu.edu.cn/。

郝兴伟

2012 年春



第2版前言

今天，Internet 已经成为一种最基本的社会基础设施，它几乎渗透到了现代社会的每一个角落。无论是 IT 专业人员、其他工作人员还是一般计算机用户，互联网已经成为人们最主要的通信、获取信息和发布信息的媒体。互联网应用的普及推动了人们对学习和了解 Internet 相关技术的社会需求。但是，走进书店或在 Internet 上查询，关于互联网的书籍铺天盖地，令人眼花缭乱。有关的书籍太多，以至于我们无所适从。为此，我想编写一本介绍互联网开发和应用的综合性书籍，使大家对目前的互联网，特别是 Web 技术，从概念、原理和应用上有一个总体的了解和把握，这就是本书第 1 版写作的初衷。

从 2005 年本书第 1 版的出版到现在，三年过去了，《Web 技术导论》一书受到了许多老师的认可，被选作教科书。我也非常高兴地收到了 20 多位任课教师的邮件，与我交流书中的相关技术，有些老师还非常诚恳地对本书提出了一些良好建议，比如：增加有关 Web 服务、SOA 等最新 Web 概念的内容，去掉操作性的多媒体制作章节，等等。这些良好的建议和这几年来我在 Web 开发中的一些新的认识和体会促使自己决定对第 1 版进行彻底的修订，增加更多新技术的讲解，特别是 Web 环境下的软件体系结构、开发模式、设计模式、AJAX 技术等新的内容，从而使本书能够紧跟互联网的发展步伐。

本书作为导论性质的书籍，将全面介绍互联网的发展历史、最新的科学进展、Web 的工作原理、计算模式和软件体系结构的演变、Web 核心技术、互联网语言、Web 设计模式、Web 客户端开发、Web 服务端开发等内容。相信这样的内容安排对大多数读者都会有所帮助。如果你是一个初学者，这本书会为你答疑解惑；如果你是一个初级的开发人员，这本书可以为你建立一个 Web 开发的基本框架，引领你进入 Web 开发的广阔天地；如果你是一个高级开发人员，本书的综合性内容也会为你阅读其他专业知识做一个基本知识的铺垫。

本书与第 1 版一样，仍然分成 6 章，主要内容如下。

第 1 章　Web 基础。介绍互联网的发展和相关概念，Web 的工作机理以及 Java 技术、XML、Web 服务等 Web 核心技术；还介绍计算机软件体系架构的演变和 SOA 体系架构的思想；最后介绍 Web 2.0 和语义 Web 的发展。

第 2 章　Web 服务器的架设和管理。首先介绍操作系统和 Web 服务器的概念；然后介绍 Windows 平台下的 Web 服务器的架设和管理，主要讲解 Windows Server 平台中的 IIS，对 IIS 的讲解比较简单，易于理解；在理解 Web 服务的管理后，重点讲解 Apache Tomcat 的架设和管理以及 Web 应用的部署等，Apache 是开发 Web 应用最常用的运行平台；最后对 Web 服务器的远程管理进行讲解。

第 3 章　HTML 和 XML 基础。首先介绍标记语言的概念，介绍 HTML 的基本语法，并安排大量实例来说明每种元素的含义和使用；对 HTML 和 XML 的本质区别进行深入的分析和总结；讲解 XML 相关的规范，包括可扩展样式语言（XSL）、XML 路径语言（XPath）、XML 查询语言（XQuery）、可扩展连接语言（XLL）、XML 文档对象模型（DOM）与

简单应用程序接口(SAX),并对它们之间的关系进行总结,这些内容对大家理解以 XML 为核心的 Web 技术具有重要意义。

第 4 章 网页设计与制作。网页是 Web 应用的主要用户界面,在 HTML 和 XML 基础上,加强网页设计的讲解,包括页面功能与内容设计、页面布局设计、页面视觉设计以及页面效果设计等;然后,介绍可视化的网页制作工具 FrontPage。

第 5 章 客户端开发。首先讲解 Web 浏览器的基本工作原理,然后讲解客户端脚本程序设计语言 JavaScript、浏览器对象模型(BOM)、HTML 文档对象模型(DOM)、Web 交互的内容,增加 AJAX 技术的讲解,最后详细讲解两个综合性客户端开发实例。

第 6 章 服务端开发。首先介绍 B/S 三层结构的概念;然后重点讲解 Java 技术及其在 Web 开发中的应用,包括 Java 程序设计语言、Java Applet、JavaBeans、Servlet 服务器程序、JSP 技术以及 MVC 设计模式;在 JSP 技术中,讲解 JSP 的语法、内置对象、数据库操作、图形操作等许多实用的内容;最后,讲解在线聊天 Web 应用的整个开发过程,同时对常用的 Java 开发工具进行介绍。

作为互联网的用户和 Web 技术的开发者和实践者,同时,作为一名高校教师,虽然,我的初衷是使本书既包含广泛的理论知识,又有很好的技术内容,但是要真正地将理论和技术结合起来是很困难的。一方面是 Web 相关的技术实在太多,笔者的知识面和认识有限,加之时间仓促;再者是考虑到读者的实际应用需求非常多样,很想把一些更实用的软件代码介绍给大家,并进行讲解,但是,受到篇幅的限制,也不能如愿。

在本书写作的过程中,我要感谢我的同事巩裕伟教授,他是一名优秀的老师,总是将计算机技术深入浅出地传授给学生,受到学生的普遍欢迎。同时,他还是一位很好的程序员,编写了大量的 Java、JSP、Visual Basic 程序和数据库应用系统。另外,他还是一位出色的作者,我们合作出版过许多计算机方面的书籍。同时我要感谢我的同事焦文江老师,他对网络环境有着很深入的研究,对网络设备非常熟悉,对待工作总是认真负责。还要感谢我的学生苏雪、常跃峰、崔旭和朱岩,他们编写了大量的程序代码,祝愿他们在以后的工作和生活中一切顺利,祝愿他们取得更大的成绩。感谢孟祥旭教授、王海洋教授、马军教授、张彩明教授、徐秋亮教授、龙世立研究员,作为领导、同事和朋友,他们在学术上和事业上都给了我很大的帮助。最后,感谢山东大学研究生院的立项资助。

由于本书涉及的内容非常广泛,在深度和广度上很难做到完美,加之作者水平有限,书中肯定存在错误和不足,敬请读者批评指正。

作者 E-mail:hxw@sdu.edu.cn。

郝兴伟

2008 年暑假



第1版前言

没有哪一项技术能和今天的Internet一样发展迅速了，它对我们的工作、生活的影响面之广、影响程度之深，使得我们不能不重视它。走进书店或在Internet上查询，关于互联网的书籍铺天盖地，令人眼花缭乱。无论你是一个专业的开发人员、普通用户还是一个计算机网络生活的爱好者，有关的书籍太多了，以至于我们无所适从。

可是要找到一本自己适合的书却很难，也许你无法说清楚你要的书是什么，因为你对互联网的认识可能刚刚开始。也许你要找一本比较全面的书，可是每一个内容都是一本很厚的专业书籍。因此，笔者计划编写一本比较全面的关于互联网的书籍，取名“Web技术导论”。在这本书中，将介绍互联网的发展历史、最新的科学进展、Web的工作原理、实现技术、互联网语言、开发工具，直到Web应用的开发、网络安全等内容。这样的内容安排相信对大量的读者会有所帮助。如果你是一个初学者，这本书会为你答疑解惑，如果你是一个初级的开发人员，这本书可以为你建立一个基本的开发框架，领你进入网络开发的广阔天地。如果你是一个高级开发人员，请您选择其他更加有针对性的书籍。

本书共6章，第1章Web基础，介绍互联网的发展、Web的工作机理、相关概念、计算模式、Web新进展等内容。

第2章Web服务器的架设和管理，介绍Web服务器的架设和管理，主要讲解了Windows 2000 Server中的IIS和Apache Tomcat的架设和管理过程和方法。

第3章HTML和XML基础，介绍了两种标记语言的基本语法，并安排了大量的例子解释每种元素的含义和使用。

第4章网页及多媒体制作，在HTML和XML基础上，介绍了可视化的制作工作，包括FrontPage和Dreamweaver，同时讲解了Photoshop图像处理技术和Flash动画制作技术。

第5章客户端开发，主要介绍了客户端脚本程序JavaScript、浏览器对象、Web交互的内容，并详细讲解了两个综合性的实例。

第6章服务端开发，介绍了Jave技术、Web三层体系结构、Servlet/JSP/EJB服务端开发。比较了几种主流的开发环境，如JSP、ASP和PHP技术，同时对Java开发工具进行了介绍。

虽然，笔者的初衷是要写一本既有理论又含技术的书籍，但是要真正地将理论和技术结合起来是很困难的。一方面是时间仓促，另一方面也考虑到读者的实际应用需求。很想把一些更实用的软件代码介绍给大家，并进行讲解，但是受到篇幅的限制，也不能如愿。

这里，要感谢我的同事巩裕伟教授，他总是将计算机技术深入浅出地传授给学生，受到学生的普遍欢迎；同时，他还是一位很好的程序员，编写了大量的Jave、JSP、Visual Basic和数据库应用系统；另外，他还是一位出色的作者，我们合作写过许多计算机图书。同时，我还要感谢我的同事焦文江老师，他对网络环境有着很深入的研究，对网络设备非常熟悉，对待工作总是认真负责。还要感谢苏雪女士，她现在一家大型的网络公司工作，主要从事网络

开发,我们一起合作开发了许多 Web 应用。还要感谢刘丰宁先生,他从事网络管理和开发工作。

最后,就是要感谢 Internet 为我们提供了海量的信息和如此快速、便捷的交流平台。

由于本书涉及的内容非常广泛,在深度和广度上很难做到完美,加之作者水平有限,书中肯定存在错误和不足,请读者批评指正。

作者的 E-mail 为 hxw@sdu.edu.cn。

郝兴伟

2004 年夏

目 录

第1章 Web基础

【本章导读】

今天，互联网就像空气一样正在渗入到人们生活的每一个角落，它不断地改变着人们的工作、生活和娱乐方式，通过Internet，人们可以上网浏览网页、收发电子邮件、上网聊天、观看在线电影、网上购物以及从事各种电子商务活动；同时，随着B/S应用模式的发展，互联网技术也彻底改变了人们传统的计算机应用和开发模式，使企业、政府机构传统的计算机应用系统部署到互联网上，彻底改变着人们的工作方式。

本章首先介绍互联网和万维网的概念，讲解万维网(WWW)中Web服务器、Web客户机的概念；讲解HTTP以及Web服务器和Web客户机的通信原理，它是万维网工作的基础，也是进行Web服务器配置、学习HTML标记语言、进行Web系统开发的概念基础。然后，对万维网中的常用概念进行介绍；同时还简要介绍Internet中的核心技术；最后对Web的发展趋势、计算机应用模式的发展、SOA计算机开发模式进行介绍。

【知识要点】

1.1节：Internet的概念、发展阶段。

1.2节：万维网的概念、万维网的组成、Web服务器、Web浏览器、HTTP。

1.3节：网站、Web应用、超文本、超链接、统一资源定位地址、端口、Web 2.0、博客、微博、RSS订阅。

1.4节：Java技术、XML技术、Web服务。

1.5节：集中式计算、C/S计算机应用模式、B/S计算机应用模式、公共对象请求代理体系结构、组件对象模型、分布式组件对象模型、面向服务体系架构、语义Web。

1.1 Internet的产生与发展

1946年，第一台电子计算机“爱尼亚克”(ENIAC)在美国宾夕法尼亚大学莫尔电子工程学院诞生。这种计算技术的革命，透出了数字信息时代的第一缕曙光。随后，微电子技术和计算机技术的发展日新月异，计算机应用日益广泛。为了进一步提高计算机的使用效率，人们需要将不同的计算机连接起来，传递数据，共享资源，计算机网络诞生了。

20世纪60年代，出现了各式各样的计算机网络，来解决计算机之间的通信和资源共享

问题。1969年,美国国防部高级研究计划署(Advanced Research Project Agency,ARPA)[①]资助了一个有关广域网络的项目,开发了一个称做阿帕网(ARPAnet)的网络,它的主要思想是构建一个没有中央控制节点的计算机网络,以便使军事计算机系统在受到打击后不会因为部分毁坏而导致整个计算机网络的瘫痪。

1969年11月21日中午,6名科学家聚会于美国加利福尼亚大学洛杉矶分校的计算机实验室,观看这里的一台计算机与远在千里之外斯坦福研究所的另一台计算机联通。这是一个历史性的时刻,正像20年后《时代》周刊的评论:这些研究者根本没有想到,他们不只是连接了两台计算机,而是宣告了网络世界的到来。

1970年,ARPAnet已初具雏形,已经将加利福尼亚州大学洛杉矶分校、加州大学圣巴巴拉分校、斯坦福大学、犹他州大学四所大学的4台计算机以分组交换协议连接起来,实现了不同型号、不同操作系统、不同数据格式、不同终端的计算机之间的通信和资源共享。

1972年,ARPAnet已建成40多个网点,开发出了三项主要功能,即以后被广泛使用的电子邮件、远程登录和文件传输。1974年,著名的TCP/IP研究成功,彻底解决了不同计算机系统之间的通信问题,计算机互联的主要障碍被克服。

1975年,ARPAnet的运行管理移交给美国国防通信局(DCA)。1982年DCA将ARPAnet各站点的通信协议全部转为TCP/IP,同时ARPAnet被分成两部分,一部分作为军用,称为MILnet,另一部分作为民用。从此,ARPAnet从一个实验型网络向实用型网络转变,成为全球Internet正式诞生的标志。

如果把Internet的发展划分阶段的话,那么1969—1984年可以看成Internet的提出、研究和试验阶段,这时的Internet以ARPAnet为主干网。由于ARPAnet采用离散结构,不设中央网络控制设备,实现了网络渠道的多样性,从而减少了系统彻底崩溃的可能性,网络的生存能力得到了保证,实现了ARPA的最初构想。

后来,Internet的发展超出了任何人的想象。1984—1992年可以看做Internet的实用发展阶段。为了使全美国的科学家和工程师能够共享那些过去只有军事部门和少数科学家才能够使用的超级计算机设施,美国国家科学基金会(National Science Foundation,NSF)于1985年提供巨资建设了全美5个超级计算中心,同时建设了将这些超级计算中心和各科研机构相连的高速信息网络NSFnet。1986年,NSFnet成功地成为Internet的第二个骨干网。NSFnet对Internet的推广起到了巨大的推动作用,它使得Internet不再是仅由科学家、工程师、政府部门使用的网络,Internet进入了以资源共享为中心的实用服务阶段。以连接NSFnet的局域网数量为例,1988年7月只有170个;到1992年1月,这一数量就发展到了4500个。

1992年以后,Internet开始进入它的商业化发展阶段,Internet用户开始向全世界扩展,并以每月15%的速度迅速增长,每30分钟就有一个网络联入Internet。随着网上通信量的急剧增长,Internet开始不断采用新的技术以适应发展的需求,其主干网由政府部门资助开始向商业计算机公司、通信公司转化。

---

① 美国国防部高级研究计划署(ARPA)成立于1958年2月,又称DARPA(Defence ARPA),它是在1957年苏联发射世界第一颗人造卫星Sputnik的背景下诞生的,其目标就是负责前瞻性科研项目的开发,以确保美国在诸多技术领域的绝对领先。

在 Internet 商业化的过程中，万维网的出现使 Internet 的使用更简单、更方便，开创了 Internet 发展的新时期。1989 年，在瑞士日内瓦粒子物理研究实验室欧洲核子研究中心(CERN)工作的蒂姆·伯纳斯-李(Tim Berners-Lee)首先提出了 WWW 的概念，并且成功地开发出世界上第一个万维网服务器和第一个万维网客户机。同年底，蒂姆为他的发明正式定名为 World Wide Web(万维网)；1991 年 5 月，万维网在 Internet 上首次露面，立即引起轰动，迅速被广泛推广应用。

美国著名信息专家、《数字化生存(Bing Digital)》一书的作者尼古拉·尼葛洛庞帝(Nicholas Negroponte)教授认为：1989 年是互联网历史上划时代的分水岭，这一年出现的万维网技术给 Internet 赋予了强大的生命力，把 Internet 带入了一个崭新的时代。

在 WWW 的发展中，还有一位杰出的人物，他就是马克·安德森(Marc Andreessen)，是他改造了 Internet 的使用界面。在早期，万维网只有文字，没有图像、声音，也没有色彩。对普通用户来说，仍缺乏一种简单的使用界面。安德森在就读伊利诺斯大学时，开始在学校里的国家超级计算中心(NCSA)兼职工作，由于感觉 Internet 界面的难于使用，他和同事贝纳一起合作，经过 6 个星期的辛苦工作，在 1993 年 1 月写出了 UNIX 版的马赛克(Mosaic)浏览器。1994 年 4 月，只有 24 岁的安德森，同硅谷风险投资家吉姆·克拉克一起创立了 Mosaic 通讯公司，集中全力开发网络浏览器。Mosaic 通讯公司后更名为网景公司，1995 年上市，1998 年 11 月 24 日被世界最大的 Internet 服务提供商美国在线(AOL)收购。

1.2 Web 及其工作原理

在 Internet 中，Web 服务是最主要的服务之一，也是使用最广泛的 Internet 服务，对于普通的用户来讲，World Wide Web 就是互联网的代名词。但是，从原理上来讲，World Wide Web 和 Internet 是两个不同的概念，两者既有密切的联系，又有着根本的不同。

1.2.1 什么是 World Wide Web

1990 年，瑞士日内瓦世界上最大的粒子物理研究实验室欧洲核子研究中心(the European Organization for Nuclear Research，CERN)提出了 World Wide Web(WWW)的概念，它是 Internet 技术、超文本技术和多媒体技术相结合的产物。当时，核物理的研究是分散在不同国家进行的，各地的研究人员通过计算机网络和 Internet 进行学术交流。在 Internet 中进行信息交流还没有一种统一的手段，因此，根据交流的信息不同(如图片、文字等)需要调用不同的 Internet 服务，很不方便。1989 年 3 月，CERN 的蒂姆·伯纳斯-李开发了一个超级文本系统；1990 年底，第一个基于字符界面的 Web 客户浏览程序开发成功；1991 年 3 月，客户浏览程序开始在 Internet 上运行；1991 年底 CERN 向高能物理学界宣布了 Web 服务。

什么是 World Wide Web 呢？从万维网诞生起，人们并没有给它一个确切的定义。可以从 Internet 的构成和服务来理解 Web。从组成上讲，Internet 是由成千上万的网络通过通信线路和网络设备连接而成的，或者说是一个全球范围的网间网。在 Internet 中，分布了成千上万的计算机，这些计算机扮演的角色和所起的作用不同。有的计算机可以收发用户

的电子邮件,有的可以为用户传输文件,有的负责对域名进行解析,更多的机器则用于组织并展示本网络的信息资源,方便用户的获取。所有这些承担服务任务的计算机统称为服务器。根据服务的内容,这些服务有 Web 服务器、文件传输服务器(FTP 服务器)、E-mail 服务器、DNS 服务器以及各种应用服务器等。

所谓 Web 服务器,就是将本地的信息用超级文本组织,向用户提供在 Internet 上进行信息浏览服务的计算机。因此,可以将 World Wide Web 看做 Internet 中所有的 Web 服务器构成的网络,通过网页中的超链接,一个 Web 服务器可以指向其他的 Web 服务器,那些 Web 服务器又可以指向更多的 Web 服务器,这样一个全球范围的由 Web 服务器组成的 World Wide Web(万维网)就形成了。

1.2.2 Web 服务器

在计算机网络中,可以将计算机分为两类,即服务器和客户机。所谓服务器就是指提供网络服务的计算机。对于服务器计算机一般需要安装服务器操作系统,例如 UNIX、Windows Server 2003、Linux 等网络操作系统。在服务器上,根据功能需要安装服务器程序。所谓服务器程序,即一种侦听程序,其基本功能是侦听用户请求,为用户提供服务。与传统的用户应用程序不同,服务器程序通常没有漂亮的用户界面。常见的服务器有 Web 服务器、FTP 服务器、E-mail 服务器、DNS 服务器、DHCP 服务器、终端服务器等。一台服务器计算机可以安装一个服务器程序,也可以安装多个服务器程序。

客户机是指普通的用户计算机,通常不以提供网络服务为目的,安装的操作系统是 Windows XP 等客户机操作系统。客户机上安装的程序也是各种应用软件,例如 Word、Excel、PowerPoint 等各种办公软件,Photoshop、Flash 等各种工具软件,上网用的 Web 浏览器、MSN、QQ 等应用软件。可见,服务器和客户机的区分不仅是操作系统不同,安装的程序也不相同,服务器上要安装服务程序,客户机上则安装应用软件。同时,要使用服务器上的服务程序,需要客户机上安装相应的客户端程序或做客户端的配置。例如,Web 浏览器是 Web 服务器的客户端程序,通过 TCP/IP 设置,可以将一台计算机设置为 DNS 客户或 DHCP 客户等。

所谓 Web 服务器,就是安装了 Web 服务器的计算机,它为用户提供网页浏览服务。简单地讲,用户通过 Web 浏览器访问 Web 服务器,Web 服务器将用户要浏览的网页发送给用户的浏览器。要使一台计算机成为一台 Web 服务器,首先要在服务器上安装服务器操作系统,例如 UNIX、Windows Server 2003、Linux 等;其次,安装专门的 Web 服务程序,如 Windows Server 内置的 IIS(Internet Information Server)服务组件、Apache/Tomcat 等。关于 Web 服务器的安装和配置,参见第 2 章。

1.2.3 Web 浏览器

在计算机网络中,服务器和客户总是成对出现的,用户通过客户端程序使用服务器,或者通过特定的设置将计算机设置为特定服务的客户机。所谓 Web 浏览器(Browser)就是前面经常提到的 Web 客户端程序,用户要浏览 Web 页面必须在本地计算机上安装 Web 浏览器软件。通过在浏览器地址栏中输入 URL 资源地址,Web 服务器将把地址中指定的网页

文件发送到客户端浏览器,并在浏览器窗口中打开。

从本质上讲,Web 浏览器是一种用于网页浏览的应用软件,有两大功能:第一,Web 浏览器是 HTML 和 XML 格式的文档阅读器,它能够对网页中的各种标记进行解释显示;第二,浏览器是一种网页客户端程序的解释机,如果网页中包含客户端脚本程序,浏览器将执行这些客户端脚本代码,从而增强网页的交互性和动态效果。不同版本的浏览器都需要遵循 HTML 规范中定义的标记集,同时为了便于脚本编程,每个浏览器程序本身也提供了相应的浏览器内置对象,类似于传统软件开发中的函数库及其标准库函数。

在 Web 发展初期,浏览器程序主要分成两类。一类为以 Lynx 为代表的基于字符的 Web 客户机程序,主要在不具备图形图像功能的计算机上使用。Lynx 是由美国堪萨斯大学的 Lou Montulli 等研制的。Lynx 操作通过光标键在页面的超链接间移动,类似于搜索引擎,因此,目前 Lynx 常用于检查网站。另一类是以 Mosaic 为代表的面向多媒体计算机的 Web 客户机程序,它可以在各种类型的小型机上运行,也可以在 IBM PC、Macintosh 机以及 UNIX 操作系统软件平台上运行。目前,Web 浏览器软件产品很多,除了微软的 IE (Internet Explorer)浏览器外,常见的浏览器有 Maxthon(傲游)、Firefox(火狐狸)、Opera 等。此外,Google、360 安全卫士等也分别推出了自己的 Web 浏览器产品。

1.2.4 HTTP 概述

在万维网中,网页浏览实际上就是 Web 浏览器和 Web 服务器之间通信的过程。Web 浏览器和 Web 服务器之间通过 HTTP 进行通信,概念模型如图 1-1 所示。

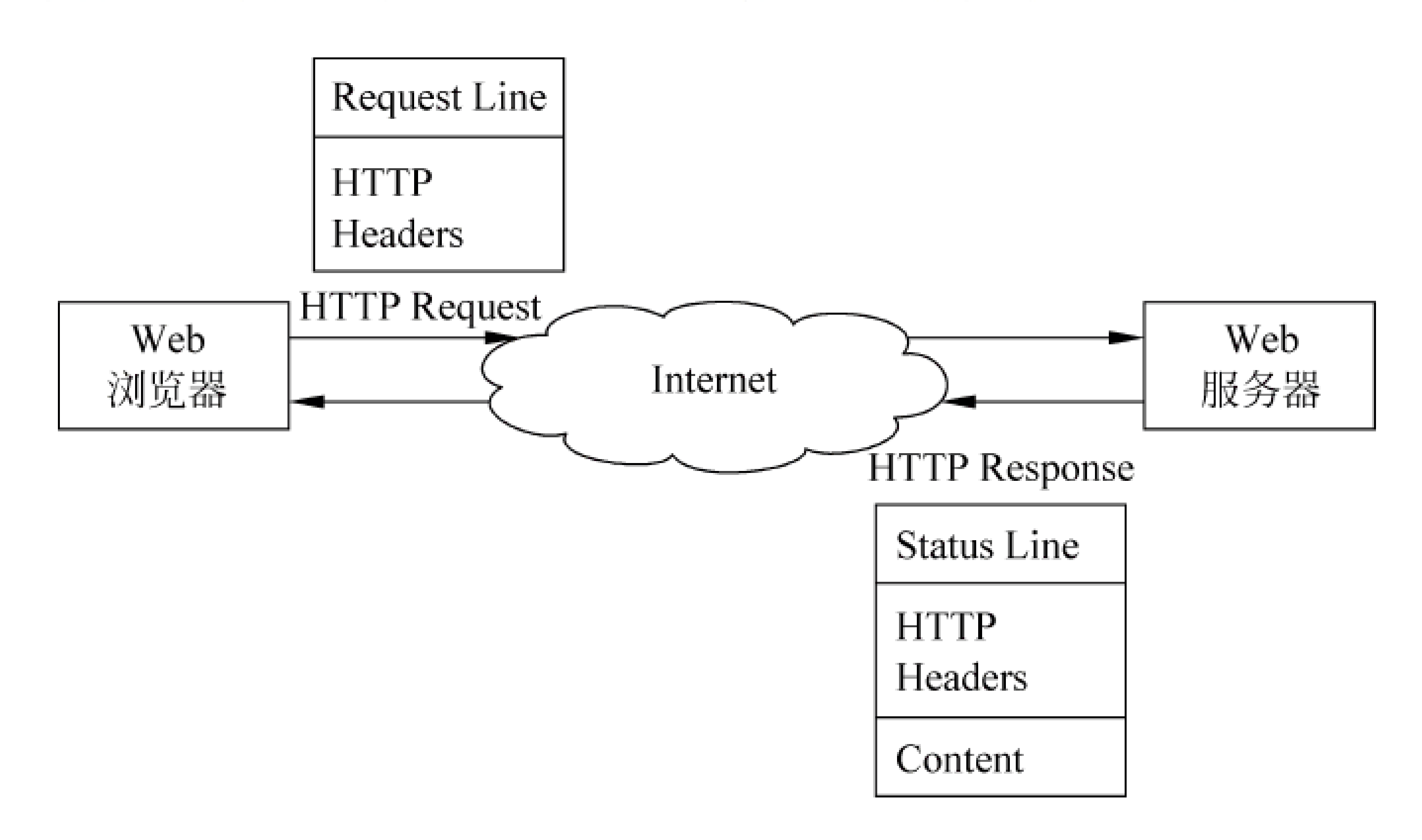


图 1-1 Web 的工作原理

1. HTTP

在 Internet 中,超文本传送协议(Hyper Text Transfer Protocol,HTTP)是应用层协议,采用请求/响应模型。当在浏览器中输入 URL 或单击页面中的一个超级链接时,浏览器就向 Web 服务器发送 HTTP 请求(HTTP Request)。HTTP 请求被送往 URL 指定的 Web 服务器,Web 服务器的 HTTP 驻留程序接收到请求后,进行必要的操作,返回 HTTP 响应(HTTP Response)给 Web 浏览器。

在 HTTP 通信中,HTTP 消息包括客户机向服务器的请求消息和服务器向客户机的响应消息。HTTP 消息由一个起始行、一个或者多个头域以及消息体(可选)组成。HTTP 头域是 HTTP 请求和响应的核心部分,它们携带关于客户端浏览器、被请求页面、服务器及其他信息。HTTP 头域包括通用头域、请求头域、响应头域和实体 4 个部分。每个头域由一个域名、冒号(:)和域值三部分组成,域名大小写无关,每个头域占一行。

(1) 通用头域:包含请求和响应消息都支持的头域。通用头域包含 Cache-Control、Connection、Date、Pragma、Transfer-Encoding、Upgrade、Via。关于各个头域的含义及取值请参考 HTTP 规范,以下同。

(2) 请求头域:允许客户端向服务器传递关于请求或者关于客户机的附加信息。请求头域包含下列字段:Accept、Accept-Charset、Accept-Encoding、Accept-Language、Authorization、From、Host、If-Modified-Since、If-Match、If-None-Match、If-Range、If-Range、If-Unmodified-Since、Max-Forwards、Proxy-Authorization、Range、Referer、User-Agent。

(3) 响应头域:允许服务器传递不能放在状态行的附加信息,这些域主要描述服务器的信息和 Request-URI 进一步的信息。响应头域包含 Age、Location、Proxy-Authenticate、Public、Retry-After、Server、Vary、Warning、WWW-Authenticate。

(4) 实体:请求消息和响应消息都可以包含实体,实体一般由实体头(Entity Header)和实体(Entity Body)组成。实体头包含关于实体的元信息,实体头包含 Allow、Content-Base、Content-Encoding、Content-Language、Content-Length、Content-Location、Content-MD5、Content-Range、Content-Type、Expires、Last-Modified 及 extension-header。extension-header 机制允许客户端定义新的实体头,但是这些域可能无法被接收方识别。实体可以是一个经过编码的字节流,它的编码方式由 Content-Encoding 或 Content-Type 定义,它的长度由 Content-Length 或 Content-Range 定义。在通信过程中,如果存在不支持的通用头域,一般将会作为实体头域处理。

2. 请求消息

当用户在浏览器地址栏中输入一个 URL 或单击页面中的超链接时,浏览器向 Web 服务器发送 HTTP 请求消息。HTTP 请求消息由两部分构成,第一行为 Request Line,后面的部分为 HTTP 头。其中,Request Line 由三部分组成,一般格式如下:

```
Method Request - URI HTTP - Version
```

(1) Method:表示对于 Request-URI 完成的方法,这个字段是大小写敏感的,包括 GET、POST、OPTIONS、HEAD、PUT、DELETE、TRACE。其中 GET 和 POST 方法使用较多,用于编码和传送变量名/变量值对参数,两者主要具有如下区别。①GET 方法使用 MIME 类型文本的格式传递参数,附加参数还能被认为是一个查询字符串。在客户端,GET 方式通过 URL 提交数据,数据在 URL 中可以看到,GET 方法提交的数据最多只能有 1024B。②POST 方法参数同样被 URL 编码,但是,变量名/变量值不作为 URL 的一部分被传送,而是放在 HTTP 请求消息内部被传送;POST 方法提交的数据多少没有限制,通常用于提交表单数据,即在<form>标记中设置 method="POST"。

(2) Request-URI：遵循 URI 格式，在此字段为星号(＊)时，说明请求并不用于某个特定的资源地址，而是用于服务器本身。

(3) HTTP-Version：表示支持的 HTTP 版本，例如 HTTP/1.1。

在 Request Line 的后面是 HTTP 头域，请求头域允许客户端向服务器传递关于请求或者关于客户机的附加信息。

一个典型的 HTTP 请求消息如下：

```
GET /2010/xxjj.html HTTP/1.1
Host:www.sdu.edu.cn
User - Agent:Mozilla/5.0 (Windows NT 5.1: rv:10.0.2) Gecko/20100101 Firefox/10.0.2
Accept: text/html,application/xhtml + xml,application/xml;q = 0.9, * / * ;q = 0.8
Accept - Language:zh - cn,zh:q = 0.5
Accept - Encoding:gzip,deflate
Connection: keep - alive
Referer: http://www.sdu.edu.cn/
Cookie:lzstat _ uv = 2061275203206613369 | 2684626; lzstat _ ss = 1170560923 _ 0 _ 1331481597
_2684626
```

上例第一行表示 HTTP 客户端通过 GET 方法获得指定 URL 下的文件。通过下面的 Referer 可以得到完整的 URL 为 http://www.sdu.edu.cn/2010/xxjjhtm。该 HTTP 请求的头域设置解释如下。

(1) Host：Host 头域指定请求资源的主机和端口号，表示请求 URL 的原始服务器或网关的位置。HTTP/1.1 请求必须包含主机头域，否则系统会以 400 状态码返回。

(2) User-Agent：User-Agent 头域包含发出请求的用户信息。

(3) Accept：告诉 Web 服务器浏览器可以接收的介质类型，＊/＊ 表示任何类型，type/＊ 表示该类型下的所有子类型(type/sub-type)。

(4) Accept-Language：浏览器声明自己接收的语言。

(5) Connection 请求：如果取值为 close，则告知 Web 服务器或者代理服务器，在完成本次请求的响应后，断开连接，不要等待本次连接的后续请求了；如果取值为 keepalive，则告知 Web 服务器或者代理服务器，在完成本次请求的响应后，保持连接，等待本次连接的后续请求。

(6) Referer 头域：Referer 头域允许客户端指定请求 URL 的源资源地址，这可以允许服务器生成回退链表，可用来登录、优化 Cache 等。它也允许废除的或错误的连接由于维护的目的被追踪。如果请求的 URL 没有自己的 URL 地址，Referer 不能被发送。如果指定的是部分 URL 地址，则此地址应该是一个相对地址。

3. 响应消息

HTTP 响应消息由状态行、头域和内容组成。响应消息的第一行为状态行，格式为：

```
HTTP - Version Status - Code Reason - Phrase
```

其中，HTTP-Version 表示支持的 HTTP 版本，例如为 HTTP/1.1；Status-Code 是一个三个数字的结果代码，用于机器自动识别；Reason-Phrase 给 Status-Code 提供一个简单

的文本描述,用于帮助用户理解。

Status-Code 编码由三个数字组成,第一个数字定义响应的类别,后两个数字没有分类的作用。常见的状态码分为五种。①1xx:信息响应类,表示接收到请求并且继续处理。②2xx:处理成功响应类,表示动作被成功接收、理解和接受。③3xx:重定向响应类,为了完成指定的动作,必须接受进一步处理。④4xx:客户端错误,客户请求包含语法错误或者是不能正确执行。⑤5xx:服务端错误,服务器不能正确执行一个正确的请求。

在状态行的后面是响应头域,响应头域允许服务器传递不能放在状态行的附加信息,这些域主要描述服务器的信息和 Request-URI 进一步的信息。如果存在不支持的响应头域,一般将会作为实体头域处理。

对于上面的 HTTP 请求,返回的一个典型的 HTTP 响应消息如下:

```
HTTP/1.1 304 Not Modified
Date: Sun,11 Mar 2012 08:28:32 GMT
Server: Apache/2.2.17(UNIX)
Connection: Keep-Alive
Keep-Alive: timeout=5,max=100
Etag:"1684db-1536-4ba3ce4300900"
```

上例第一行表示 HTTP 服务端响应一个 GET 方法。下面是头域,解释如下。

(1) Server:Server 响应头域包含处理请求的原始服务器的软件信息。此域包含多个产品标识和注释,产品标识一般按照重要性排序。

(2) Connection 响应:和 Connection 请求对应。如果取值为 close,则表明连接已经关闭。如果取值为 keepalive,则表明连接保持着,在等待本次连接的后续请求。

(3) Keep-Alive:如果浏览器请求保持连接,该头部表明 Web 服务器保持连接的时间(秒)。

(4) Etag:一个 URL 对象的标识,作用与 Last-Modified 类似,主要供 Web 服务器判断一个对象是否改变。例如,前一次请求某个 html 文件时,获得其 Etag,当再次请求同一个文件时,浏览器就会把先前获得的 Etag 值发送给 Web 服务器,然后 Web 服务器会把这个 Etag 跟该文件的当前 Etag 进行对比,以判断文件是否已经改变。

4. 实体

HTTP 请求消息和 HTTP 响应消息都可以包含实体信息,实体信息一般由实体头和实体组成。实体可以是一个经过编码的字节流,它的编码方式由 Content-Encoding 或 Content-Type 定义,它的长度由 Content-Length 或 Content-Range 定义。extension-header 允许客户端定义新的实体头,但是这些域可能无法被接收方识别。

在 HTTP 规范中,获取 HTTP 通信信息,可以分析 Web 服务器和 Web 浏览器的配置。在 Firefox 浏览器中,可以安装 LiveHTTPHeaders 插件来抓取 http,插件的下载网址为 http://livehttpheaders.mozdev.org/installation.html。下载该插件,安装 Firefox 浏览器,选择"工具"→"附加组件"命令,安装 LiveHTTPHeaders 插件,则在"工具"菜单中添加 LiveHTTPHeaders 命令,执行该命令打开抓包窗口,然后在地址栏中输入一个网址,则在 LiveHTTPHeaders 窗口可看到每个 URL 链接的 HTTP 请求和响应消息。

1.3 概念及术语

在 Web 中,新的概念和新的术语很多,随着 Web 应用的普及,这些本来是专业的概念和术语已经大众化了。下面从计算机专业的角度对 Web 中的一些常用概念进行简要介绍。

(1) 网站(Web Site)。网站又称 Web 站点,是 Internet 中提供信息服务的机构,这些机构的计算机连接到 Internet 中,向用户提供 Web 服务。此外,随着 B/S 架构的兴起,网站已经成为计算机应用的主要模式。例如,各种各样的电子商务平台,其作用已经不再是单纯的信息服务,而是一个计算机应用系统了。

从技术上讲,一个 Web 站点由一个主目录、子目录及其包含的网页文件、图片文件及其他各类文件以及相关的数据库构成。网页文件通常包含客户端脚本程序和服务端脚本程序,并通过超链接连接在一起,形成特定的应用逻辑,构成一个特定的 Web 应用。因此,网站又称为 Web 应用。

Web 应用和传统的应用程序相比,有两点主要的不同。①程序的构成不同。传统的应用程序通常是由一个 exe 文件和相关的 dll 库构成,而 Web 应用则是由一个主目录及其包含的子目录和大量网页文件构成。②运行环境不同。传统应用程序在操作系统上运行;而 Web 应用中网页中的程序包含服务端的脚本程序和客户端脚本程序,服务端的脚本程序在 Web 服务器上运行,客户端脚本程序在 Web 浏览器中运行。

(2) 超文本(Hypertext)。超文本是一种文本显示与链接技术,可以对文本中的有关词汇或句子建立链接(即超链接),使其指向其他段落、文本或链接到其他文档。通过超链接,可以在文档之间、文档内部之间跳转,这种文本的组织方式与人们的思维方式和工作方式比较接近。

当超文本显示时,建立了链接的文本、图片通常以下划线、高亮等不同的方式显示,来表明这些文本或图片对应一个超链接。当鼠标指针移过这些文字时,指针会变成手形,单击超链接文本或图片,可以转到相关的位置。

(3) 超级链接(Hyperlink)。在 Web 页中当用户单击它时可以转到其他 Web 页或当前页面的其他地方的文字、图片等对象,即该超级链接是文本超链接和图片超链接。如果是文本超链接,超链接在 Web 页上往往带有下划线或增亮显示。当用户将鼠标指针指向一个超链接时,指针会改变为手的形状。

(4) Web 页(Web Page)。Web 页是指 Web 服务器上的一个个超文本文件,或者是它们在浏览器上的显示屏幕。Web 页中往往包含指向其他 Web 页面的超级链接。在 Web 页面中,除了文本、图片等内容外,通常还包含客户端或服务端脚本程序。因此,Web 页面可分为普通的 htm 页面和服务器页面(Server Page)。含有服务端脚本程序的页面称为服务器页,根据程序的语言类型,有 JSP 页面(Java Server Page)、ASP 页面(Active Server Page)等。不含有服务端脚本程序的页面即为普通 htm 页面。普通 htm 页面可包含客户端脚本程序,如 JavaScript 程序。

(5) 主页(Home Page)。对于一个 Web 站点,主页通常是站点的第一个 Web 页,这类似于传统应用程序的主窗口。主页中往往列出了网站的信息目录,或指向其他站点的超链接,主页是一个网站的入口。当用户访问一个网站时,如果在 URL 中不指定特定的网页文

件，则 Web 服务器将站点主页发送到客户端。在 Web 服务器的配置中，主页又称为默认文档。

(6) 统一资源定位地址(Uniform Resource Locator，URL)。统一资源定位地址可以唯一标识一个 Web 页、网页中的一个图片或 Internet 上其他资源的一个地址，它将 Internet 提供的各类服务统一编址，以便用户通过 Web 客户浏览程序进行信息查询。URL 的一般形式为：

```
资源类型://网址[:端口号][/[文件路径/文件名]][?参数名=参数值&参数名=参数值…]
```

在 URL 中，除了网址外，其他内容都是可选的。其中，服务类型主要包括 http、ftp 等；网址即服务器的域名或 IP 地址，端口号对应一个特定的服务，默认端口号可以省略，例如 Web 服务的默认端口为 80，FTP 服务的默认端口为 21 等；文件路径为网页相对于主目录的相对路径；文件名是用户浏览器指定的要下载的网页文件；如果有参数，在文件名后面跟字符“?”列出参数名/参数值对，不同的参数名/参数值对之间用“&”分开。

在浏览器地址栏中，默认端口号可以省略不写，如果不指定文件路径和文件名，则默认访问站点根目录下的主页文件，主页文件由 Web 服务器指定。例如，如果用户在浏览器地址栏中输入的 URL 为 http://www.sdu.edu.cn/，则表明用户要下载域名为 www.sdu.edu.cn 的 Web 服务器中根目录下的主页文件。

(7) 端口(Port)。在计算机通信中，端口用于识别一个唯一的通信程序。根据 OSI 参考模型的规定，数据通信最终被封装成包在 Internet 中传递，当数据包到达接收方时，在接收方的计算机上可能运行着多个服务程序，服务程序将根据到达的数据包中的端口号来决定是否接收一个到达的数据包，这和旅客在机场的行李传送带旁提取自己的包裹类似。

在 TCP/IP 中，按协议类型划分，端口可以分为 TCP、UDP、IP 和 ICMP(Internet 控制消息协议)等。其中 TCP 端口和 UDP 端口是最常见的端口类型。按照端口号分布划分，端口分为知名端口(Well-Known Ports)和动态端口(Dynamic Ports)两部分。所谓知名端口，是指范围从 0 到 1023 的端口，这些端口号一般固定分配给一些知名的公共服务。例如，FTP 服务的端口号为 21，SMTP 服务的端口号为 25，HTTP 服务的端口号为 80 等。动态端口是指范围 1024～65 535，它是操作系统为通信程序临时分配的端口。例如，为浏览器分配的端口，当关闭浏览器窗口时，端口被收回。

在网络安全中，木马和黑客程序均是通过特定的端口来控制计算机的，因此，通过设置 TCP/IP 筛选可以很容易地切断黑客或木马的攻击。利用命令行命令“netstat -a -n”查看当前系统正在进行通信的协议端口，也可以安装“360 安全卫士”，查看当前的通信进程及所使用的端口号。这有助于发现系统是否有木马在运行。

(8) Web 2.0。回想 Web 诞生之初，人们面对一个个静态的网页，已经兴奋不已。今天，人们习惯地把这个时期(2003 年以前的互联网模式)的互联网称为 Web 1.0，这是一个信息消费的时代，人们通过浏览器获取信息。在 Web 1.0 时代，Netscape 脱颖而出，成为互联网耀眼的新星，它的浏览器，把广大的普通用户带入了互联网。同时，Yahoo 公司提出了互联网黄页，Google 公司推出了深受欢迎的搜索服务，它们为互联网的发展做出了巨大的贡献。

随着网络的发展，网站的拥有者发现，只有网民的参与，才能持久地提高与保持网站的人气。从一开始出现的“论坛”到快速火热起来的“博客”，互联网事实上已经逐渐开始了一

种理念上的转变，实践着从 Web 1.0 到 Web 2.0 的跨越。关于 Web 2.0，并没有一个统一的定义，它通常是指注重用户的交互作用，强调用户的广泛和深入参与，被认为是下一代的软件设计模式和商业模式。Web 2.0 理念使得网站的展现形式更加多样化，产生了很多的典型产品，例如论坛、名人博客等。

(9) 博客(Blog)。博客(Blog)的全名是 Web log，即“网络日志”，后来缩写为 Blog。Blog 是继 E-mail、BBS、ICQ 之后出现的第四种网络交流方式，是以超级链接为武器的网络日记，代表着一种新的生活方式和新的工作方式。而博客(Blogger)则指的是使用特定的软件，在网络上出版、发表和张贴个人文章的人。

从技术上讲，一个 Blog 就是一个网页，它通常是由简短且经常更新的帖子所构成，这些张贴的文章按照年份和日期倒序排列。Blog 的内容和目的有很大的不同，有的是个人的一些评论、随笔、日记、照片等，有的则是一群人基于某个特定主题的集体创作。2002 年 8 月，“博客中国”(http://www.blogchina.com/)网站开通，“博客”现象在中国互联网界出现。

现在，许多门户网站(例如新浪、网易等)都提供博客功能，用户可以免费注册并发表文章。某种意义上说，博客是一种新的文化现象，博客的出现和繁荣，真正凸显网络的知识价值，标志着互联网发展开始步入更高的阶段。

(10) 微博(MicroBlog)。微博，即微博客(MicroBlog)的简称，是一个基于用户关系的信息分享、传播以及获取平台，用户可以通过 Web、WAP 以及各种客户端软件来更新或获取信息，以 140 字左右的文字更新信息，并实现即时分享。最早也是最著名的微博是美国的 Twitter。2009 年 8 月，新浪网推出“新浪微博”，提供微博服务，微博正式进入中文上网主流人群视野。

与博客相比，微博是一种通过关注机制分享简短实时信息的广播式的社交网络平台，具有可以单向或双向关注机制、内容简短、实时广播的特点。2010 年开始，微博像雨后春笋般崛起，四大门户网站均开设微博服务。据中国互联网络信息中心(CNNIC)发布的《第 28 次中国互联网络发展状况统计报告》显示，2011 年上半年，中国微博用户从 6331 万增至 1.95 亿，增长约 2 倍。微博在网民中的普及率从 13.8%增至 40.2%，手机微博在网民中的使用率从 15.5%上升到 34%。

(11) RSS 订阅。网络信息量每天都以惊人的速度增长，人们通常以搜索引擎的方式搜索需要的信息，对于扑面而来的新闻，则需要花费大量的时间冲浪和从新闻网站下载。即使如此，有大量的新闻或信息可能因为没有及时浏览而错过。

相对于传统的信息浏览，RSS(Really Simple Syndication，简易信息聚合)订阅则是一种全新的资讯传播方式，它采用推技术将订阅的页面发送到客户的 RSS 阅读器或 Web 浏览器中。只要用户下次打开 RSS 阅读器或支持 RSS 订阅的浏览器，被订阅的页面将显示在频道列表中。提供 RSS 订阅的站点或页面通常被标记为 XML 或 RSS 的橙色图标，例如，网易 RSS 订阅中心(http://www.163.com/rss/)。

1.4 Web 相关技术

进入 20 世纪 90 年代以后，随着 Internet 技术的不断发展，特别是 Web 的出现，对计算机的计算模式，软件开发模式、应用模式都产生了重要的影响，这导致了一系列相关技术的

出现,并推动着 Web 技术的发展。

1.4.1 Java 技术

Java 技术是 Sun 公司于 1995 年推出的一种极富创造力的计算平台。狭义上讲,Java 技术可以理解为 Java 语言;广义上讲,Java 技术包括 Java 语言、Java 虚拟机以及 Java API 等。Java 技术为用户带来了无数令人兴奋的可能性,它几乎使所有应用程序(包括游戏、工具和服务程序)能在任何计算机或设备上运行。Java 技术的多功能性、有效性、平台的可移植性以及安全性已经使它成为网络计算领域最完美的技术。今天,Java 技术已经无处不在,从桌面 PC 到科学超级计算机和互联网,从移动电话到移动手持设备,从家庭游戏机到信用卡,几乎在所有的网络和设备上都会看到 Java 技术的身影。

1. Java 的出现

1991 年,Sun 公司计划开拓消费类电子产品市场,为电视、烤面包箱等家用消费类电子产品开发一个分布式代码系统,目的是可以通过 Internet 与家电产品进行交互,以便对其进行控制。Sun 公司内部人员把这个项目称为 Green,该小组的领导人是 James Gosling,他是一位非常杰出的程序员。Gosling 于 1984 年加盟 Sun 公司,之前在一家 IBM 研究机构工作,他是 SunNeWs 窗口系统的总设计师,也是第一个用 C 实现的 EMACS 文本编辑器 COSMACS 的开发者。

开始,他们准备用 C++语言开发,但是,C++太复杂,且存在安全性问题。于是,在 1991 年 6 月 James Gosling 开始准备基于 C++开发一个新的语言,看着窗外的一棵老橡树,就将这个新的语言命名为 Oak,它就是 Java 的前身。Oak 是一种用于网络的精巧而安全的语言,Sun 公司使用它参加一个交互式电视项目的投标,结果败于 SGI 公司①,为此 Oak 几乎销声匿迹。此时,受到 Marc Andreessen 开发的 Mosaic 和 Netscape 的启发,他们将 Oak 继续完善。因为,此时发现在此之前 Oak 已是 Sun 公司另一个语言的注册商标,他们将新的 Oak 改名为 Java,即太平洋上一个盛产咖啡的岛屿(爪哇岛)的名字。

James Gosling 在开始写 Java 时,并不局限于扩充语言机制本身,更注重于语言所运行的软硬件环境。他要建立一个系统,这个系统运行于一个巨大的、分布的、异构的网格环境中,完成各种电子设备之间的通信与协同工作。James Gosling 在设计中采用了虚机器码(Virtual Machine Code)方式,即 Java 语言编译后产生的是虚拟机,虚拟机运行在一个解释器上,每一个操作系统均有一个解释器。这样一来,Java 就成了平台无关语言,这和 James Gosling 设计的 SunNeWs 窗口系统有着相同的技术思想。在 SunNeWs 中,用户界面统一用 PostScript 描述,不同的显示器有不同的 PostScript 解释器,这样便保证了用户界面良好的可移植性。

后来,Patrick Naughton 加入到该项目,Naughton 也是 Sun 公司的技术骨干,曾经是 Open Windows 项目的负责人。整个工作进展神速,经过 17 个月的奋战,整个系统顺利完成。它是由一个操作系统、一种语言(Java)、一个用户界面、一个新的硬件平台、三块专用芯

---

① SGI 公司为美国图形工作站生产厂商。

片构成的。项目完成后，在 Sun 公司内部做了一次展示和鉴定，观众的反应是：在各方面都采用了崭新的、非常大胆的技术。

2. Java 语言环境

1994 年，WWW 已如火如荼地发展起来。Gosling 意识到 WWW 需要一个中性的浏览器，它不依赖于任何硬件平台和软件平台，它应是一种实时性较高、可靠安全、有交互功能的浏览器。于是，Gosling 决定用 Java 开发一个新的 Web 浏览器，这就是 1994 年秋天完成的 WebRunner，这个原型系统展示了 Java 可能带来的广阔市场前景。后来，WebRunner 改名为 HotJava，并于 1995 年 5 月 23 日发表后，在产业界引起了巨大轰动，Java 的地位也随之得到肯定。

1995 年，Sun 公司虽然推出了 Java，但这只是一种语言，而要想开发复杂的应用程序，必须要有一个强大的开发库支持。因此，又经过一年的试用和改进，Sun 公司在 1996 年 1 月 23 日发布了 JDK 1.0。这个版本包括了两部分：运行环境(Java Running Environment，JRE)和开发环境(Java Development Kit，JDK)。在运行环境中包括了核心 API、集成 API、用户界面 API、发布技术、Java 虚拟机(Java Virtual Machine，JVM)五个部分。开发环境包括编译 Java 程序的编译器(即 javac)。在 JDK 1.0 时代，JDK 除了抽象窗口工具包(Abstract Windowing Toolkit，AWT，一种用于开发图形用户界面的 API)外，其他的库并不完整。

Sun 公司在推出 JDK 1.0 后，紧跟着，在 1997 年 2 月 18 日发布了 JDK 1.1。JDK 1.1 相对于 JDK 1.0 最大的改进就是为 JVM 增加了 JIT(即时编译)编译器。JIT 和传统的编译器不同，传统的编译器是编译一条，运行完后再将其扔掉，而 JIT 会将经常用到的指令保存在内存中，在下次调用时就不需要再编译了，这样 JDK 在效率上有了非常大的提升。

随后，一些著名的计算机公司纷纷购买了 Java 的使用权，IBM、Apple、DEC、Adobe、Silicon Graphics、HP、Oracle、Toshiba、Netscape 和 Microsoft 等大公司相继购买了 Java 的许可证。另外，众多的软件开发商也开发了许多支持 Java 的软件产品。在以网络为中心的计算机时代，不支持 HTML 和 Java，就意味着应用程序的应用范围只能限于同质的环境。

Java 的平台无关性给未来的计算模式产生了革命性的影响，它是继 HTML 后，Internet 发展的又一个里程碑。

3. Java 的技术特征

在 Sun 公司的 Java 语言白皮书中，说明 Java 语言有如下特征：简单、面向对象、体系结构中立、解释执行、可移植、分布式、具有安全性。

(1) 简单(Simple)。主要体现在三个方面：①Java 语言风格来源于 C++，因此 C++程序员可以很快地上手；②Java 摈弃了 C++中容易引发错误的地方，如指针，增加了内存管理等一些新的特色；③Java 提供了丰富的类库，使用户编程更加简单。

(2) 面向对象(Object-Oriented)。Java 是面向对象的语言，摈弃了 C++中全局变量等与面向对象思想冲突的内容。

(3) 体系结构中立(Architecture Neutral)。一般情况下，网络环境都是异构的，如何使一个应用程序能够在不同硬件、不同操作系统平台的计算机上运行，始终是一个难题。Java

将它的程序编译成一种结构中立的中间文件格式，由 Java 虚拟机来解释执行这种中间代码。这使得 Java 应用程序可以在不同的处理器中执行，现在几乎所有的主流计算机系统都能运行 Java。

(4) 解释执行(Interpreted)。Java 解释器能直接地在任何机器上执行 Java 字节码。

(5) 可移植(Portable)。体系结构无关的特性使 Java 程序可以在配备了 Java 虚拟机的任何计算机系统上运行。另外，通过定义独立于平台的基本类型及其运算，Java 数据得以在任何硬件平台上保持一致。

(6) 分布式(Distributed)。Java 程序的程序库可以很容易地与 HTTP 和 FTP 等 TCP/IP 配合，从而使 Java 程序可以凭借 URL 打开并访问网络对象，对程序员来讲，访问方式和访问本地文件系统几乎一样，这就为 Internet 等分布环境提供内容带来了方便。

(7) 安全性(Secure)。Java 是被设计用于网络和分布式环境的，安全性自然是一个重要的考虑因素。Java 的安全性可以从两个方面考虑：①内存的安全性，如摈弃了 C++中的指针，从而避免了非法内存操作和内存泄漏；②当用 Java 来创建浏览器内容时，语言功能和浏览器本身的功能结合，使它更安全。

4. Java 的发展

十多年来，Java 技术的发展总是日新月异，从奠定 Java 根基的 Java 开发包 JDK 1.0，到今天的 JDK 6.0，Java 为开发人员提供的标准类库越来越丰富，Java 技术取得了长足的进步。

从 JDK 1.0 到 JDK 1.1.8，JDK 1.x 经过了 9 个小版本的发展，已经初具规模。1998 年 12 月 4 日，Sun 公司发布了 Java 历史上最重要的一个 JDK 版本——JDK 1.2，这个版本标志着 Java 进入了 Java2 时代，进入 Java 的飞速发展时期。

在 Java2 时代，Sun 公司对 Java 进行了很多革命性的变化，将 JDK 1.2 一分为三，Java 被分成了 J2EE(Java2 Platform Enterprise Edition)、J2SE(Java2 Platform Standard Edition)和 J2ME(Java2 Platform Micro Edition)，分别面向企业级、桌面、嵌入式和移动计算等领域。这些革命性的变化一直沿用到现在，对 Java 的发展形成了深远的影响。

从 JDK 1.2 开始，Sun 公司以平均两年一个版本的速度推出新的 JDK。2000 年 5 月 8 日，Sun 公司对 JDK 1.2 进行了重大升级，推出了 JDK 1.3。Sun 在 JDK 1.3 中同样进行了大量的改进，主要表现在一些类库(如数学运算、新的 Timer API 等)上、在 JNDI 接口方面增加了一些 DNS 的支持、增加了 JNI 的支持等。2002 年 2 月 13 日，Sun 发布了 JDK 历史上最为成熟的版本 JDK 1.4。这次 Sun 公司将主要精力放到了 Java 的性能上，使 JDK 1.4 的性能有了质的飞跃。到 JDK 1.4 为止，人们已经可以使用 Java 实现大多数的应用了。

虽然从 JDK 1.4 开始，Java 的性能有了显著的提高，但 Java 又面临着另一个问题，那就是复杂。2004 年 10 月，Sun 公司发布了 JDK 1.5，同时，将 JDK 1.5 改名为 J2SE5.0。与 JDK 1.4 不同，JDK 1.4 的主题是性能，而 J2SE5.0 的主题是易用。

2006 年 4 月，Sun 公司推出 J2SE6.0 测试版，2006 年 12 月，代号为 Mustang(野马)的 J2SE6.0 正式版推向市场，在性能、易用性方面得到了前所未有的提高。2006 年 12 月，Sun 公司发布 JRE6.0。

2009 年 4 月 20 日，Oracle 和 Sun 公司发布了联合声明，Oracle 收购了 Sun 公司。Sun，这个让全球软件开发者曾热血沸腾、视为心灵家园的品牌，这个为世界贡献了一整套包括

Java 在内的全系列开源软件和“网络即计算机”战略方向的 Sun，将为 Oracle 公司贡献出自己的所有，也将慢慢地淹没在历史的长河中……

1.4.2 XML 技术

可扩展置标语言(eXtensible Markup Language，XML)是 Internet 上最具权威的数据表示和数据交换标准，它是国际标准化组织(International Organization for Standardization，ISO)的通用标记语言标准(Standard for General Markup Language，SGML)的一个简化子集。

1. XML 的技术特征

XML 关注信息本身，是 Web 上表示结构化信息的一种标准文本格式。与传统的注重页面信息显示的超文本标记语言(Hypertext Markup Language，HTML)相比，关注于内容的 XML 具有以下优点：良好的可扩展性，语言简单有效，可自行定义标记；内容与形式的分离，主要刻画数据内容，不考虑显示效果；有严格的语法要求，便于分析和与数据库信息转换；便于传输，为纯文本形式，可通过 HTTP 直接传输，可跨越防火墙等。

XML 的出现和发展对于 Internet 产生了巨大的影响，如果说 Java 实现了代码的平台无关性，那么 XML 则实现了数据的平台无关性。今天，XML 已经逐渐成为整个 Web 的基本结构和未来各种发展的基础，由于 XML 能针对特定的应用定义自己的标记语言，使得 XML 可以在电子商务、政府部门、各行业领域提供各具特色的独立解决方案。同时，XML 作为一种通用的数据交换语言，已经成为业界的一种具有垄断性的标准，在跨平台跨系统数据交换方面拥有无可比拟的优势，其在企业级开发中所扮演的角色越来越重要。但是，和关系数据库拥有强大的存储和分析引擎不同，XML 只专注于数据的表示，这也使得 XML 在数据量急速膨胀时，如何有效地管理和使用 XML 成为了一件令人头痛的事情。

2. XML 相关技术标准

虽然 XML 标准本身相对简单，但与 XML 相关的标准却种类繁多，W3C 制定的相关标准就有二十多个，采用 XML 制定的各种应用标准也很多。除了标准种类繁多外，标准之间通常还互相引用，特别是应用标准，它们的制定不仅仅使用的是 XML 标准本身，还常常用到了其他很多标准。在 XML 标准体系中，XML 相关标准可分为元语言标准、基础标准、应用标准三个层次。

(1) 元语言(Meta-language)标准：描述的是用来描述标准的元语言，是整个体系的核心，其他 XML 相关标准都是用它制定的或为其服务的。

(2) 基础标准(Foundation Standards)：这一层次的标准是为 XML 的进一步实用化制定的，规定了采用 XML 制定标准时的一些公用特征、方法或规则。例如，XML Schema 描述了更加严格地定义 XML 文档的方法，以便可以更自动化地处理 XML 文档；XML Namespace 用于保证 XML DTD 中名字的一致性，以便不同的 DTD 中的名字在需要时可以合并到一个文档中。

(3) 应用标准(Application Standards)：以 XML 为基础制定的行业标准。比较常用的应用标准包括 SVG(有关矢量图形)、SMIL(有关多媒体同步显示)、MathML(有关数学公式符号)等。在电子商务领域的应用标准有 Micropayments(W3C 制定)、BizTalk

(Microsoft 发起的电子商务 Schema 库)、ebXML(联合国 UN/CEFACT 小组和 OASIS 共同发起)、PIP(由诸多 IT 业的巨子组成的一个标准化组织 RosettaNet 的应用网络标准)等。

XML 相关技术标准体系如图 1-2 所示。

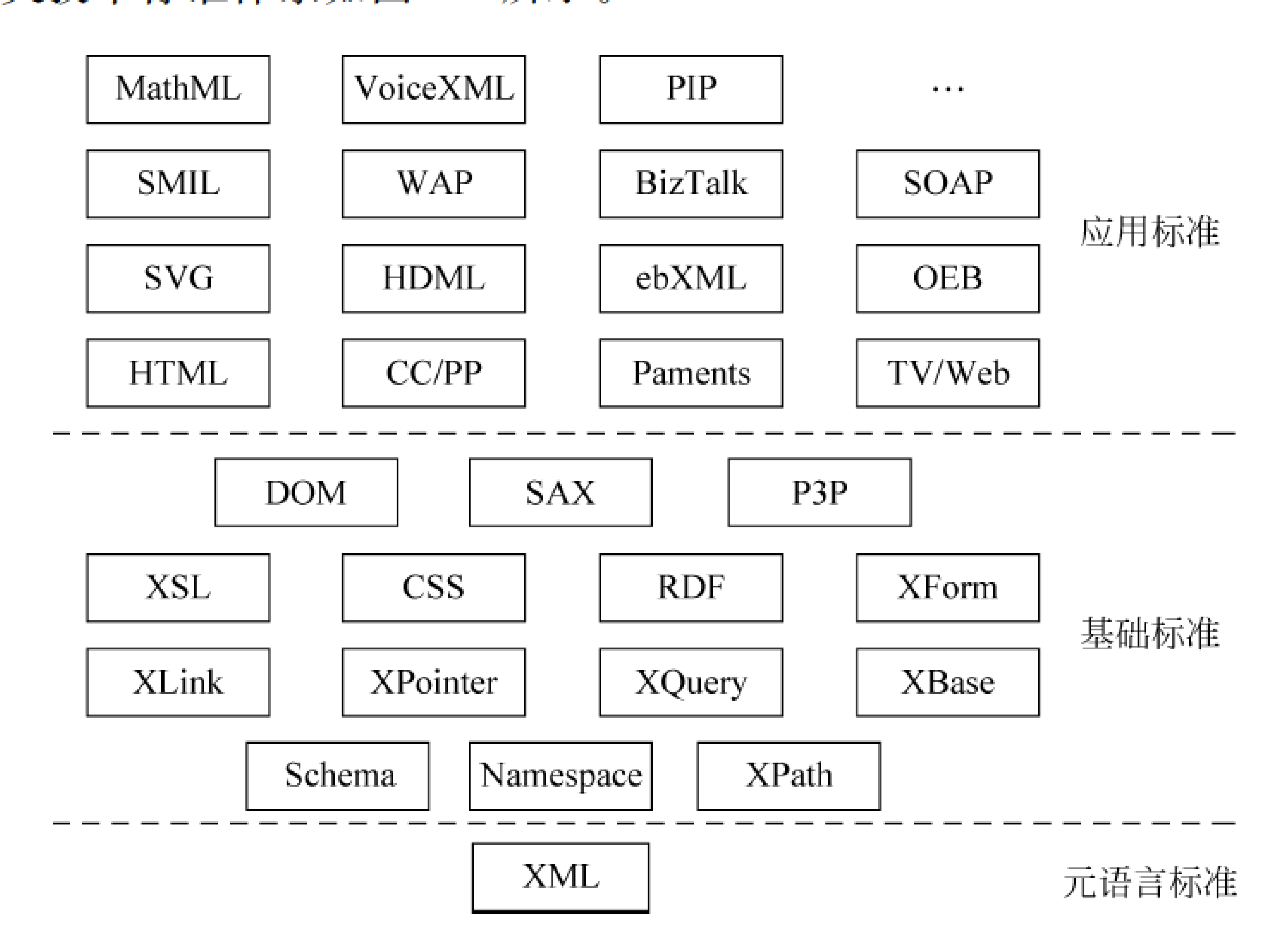


图 1-2 XML 技术标准体系

在 XML 技术标准体系中,主要的基础标准的功能划分是:XML Schema 描述了更加严格定义 XML 文档的方法,以便可以更自动地处理 XML 文档;XML Namespace 用于保证 XML DTD 中名字的一致性,以便不同的 DTD 中的名字在需要时可以合并到一个文档中;XPath 描述如何识别、选择、匹配 XML 文件中的各个构成元件,包括元素、属性、文字内容等;XPointer 和 XLink 标准,规定了有关定位、链接方面的内容;XQuery 的目的是为从 Web 文档中提取数据,提供一种灵活的查询机制;XSLT 则实现文档格式转换,主要是将 XML 转换为 HTML 格式进行显示;CSS 也是用来作为 XML 文档显示的样式标准;DOM 定义了一组与平台和语言无关的接口,以便程序和脚本能够动态访问和修改 XML 文档内容、结构及样式。

1.4.3 Web 服务

在传统意义下,软件开发模式是编程人员编写系统的每一行代码,当然包括库函数调用,调用库函数可以提高编程效率和代码质量。软件开发完成后,用户需要安装,才能使用。Web 服务(Web Service)是一种新型的软件开发模式,在该模式下,传统的软件功能模块不再以函数方式提供以实现二进制代码级的重用,而是被封装成 Web 服务,实现业务级的重用和集成。

1. 基于 Web 服务的软件开发模式

假设开发人员需要搭建一个商务网站,这个网站需要一个验证客户合法身份的功能。可以采取以下方法实现这个功能。

（1）由开发人员自己编写安全验证所需的全部程序代码。

（2）购买这段程序（通常是一个ActiveX组件）。在收到组件之后，首先将组件注册在自己的机器上，然后根据组件类型库产生接口文件。在实际编程中就可以使用这个接口文件来访问组件服务。

（3）使用Web服务。只需在自己的程序中通过访问某个服务的URL地址，得到一份XML描述，并使用这个描述文件产生一个接口文件。然后，在实际编程中，通过这个接口文件来访问服务。该模式，这个服务并不运行在用户的机器上，而是运行在服务提供者的服务器上。

基于Web服务的软件开发模式如图1-3所示。

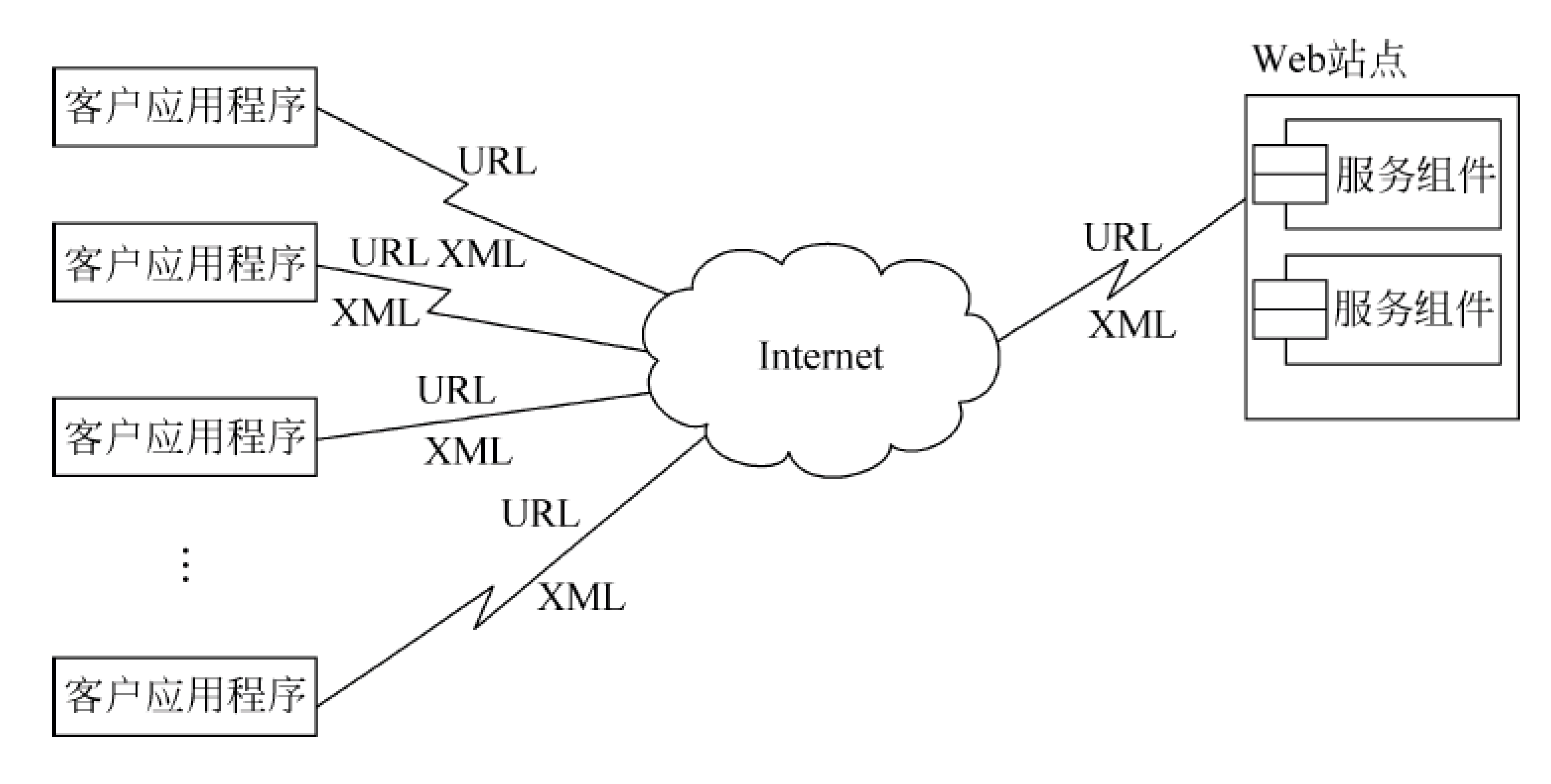


图1-3　基于Web服务的软件开发模式

Web服务使用标准化的XML消息传递机制作为基本的数据通信方式，消除使用不同组件模型、操作系统和编程语言的系统之间存在的差异，使异类系统能够作为计算网络的一部分协同运行。开发人员可以使用像过去创建分布式应用程序时使用组件的方式，创建由各种来源的Web服务组合在一起的应用程序。

2. Web服务体系架构

Web服务是一种自包含、自解释、模块化的在线应用程序。采用XML作为基本的标记语言，SOAP（Simple Object Access Protocol，简单对象访问协议）作为互操作协议，WSDL（Web Service Description Language，Web服务描述语言）作为服务描述语言，通过UDDI（Universal Description Discovery & Integration，通用描述、发现与集成）可以把服务注册到互联网以便使用者搜索，为用户提供服务。

Web服务的体系结构由三个参与者和三个基本操作构成。三个参与者分别是服务提供者、服务请求者和服务代理，三个基本操作分别为发布（publish）、查找（find）和绑定（bind），其关系如图1-4所示。

服务提供者将其服务发布到服务代理的一个目录上；当服务请求者需要调用该服务时，服务提供者首先利用服务代理提供的目录去搜索该服务，得到如何调用该服务的信息，然后根据这些信息去调用服务提供者发布的服务；当服务请求者从服务代理得到调用所需服务的信息之后，通信是在服务请求者和服务提供者之间直接进行，而无须经过服务代理。

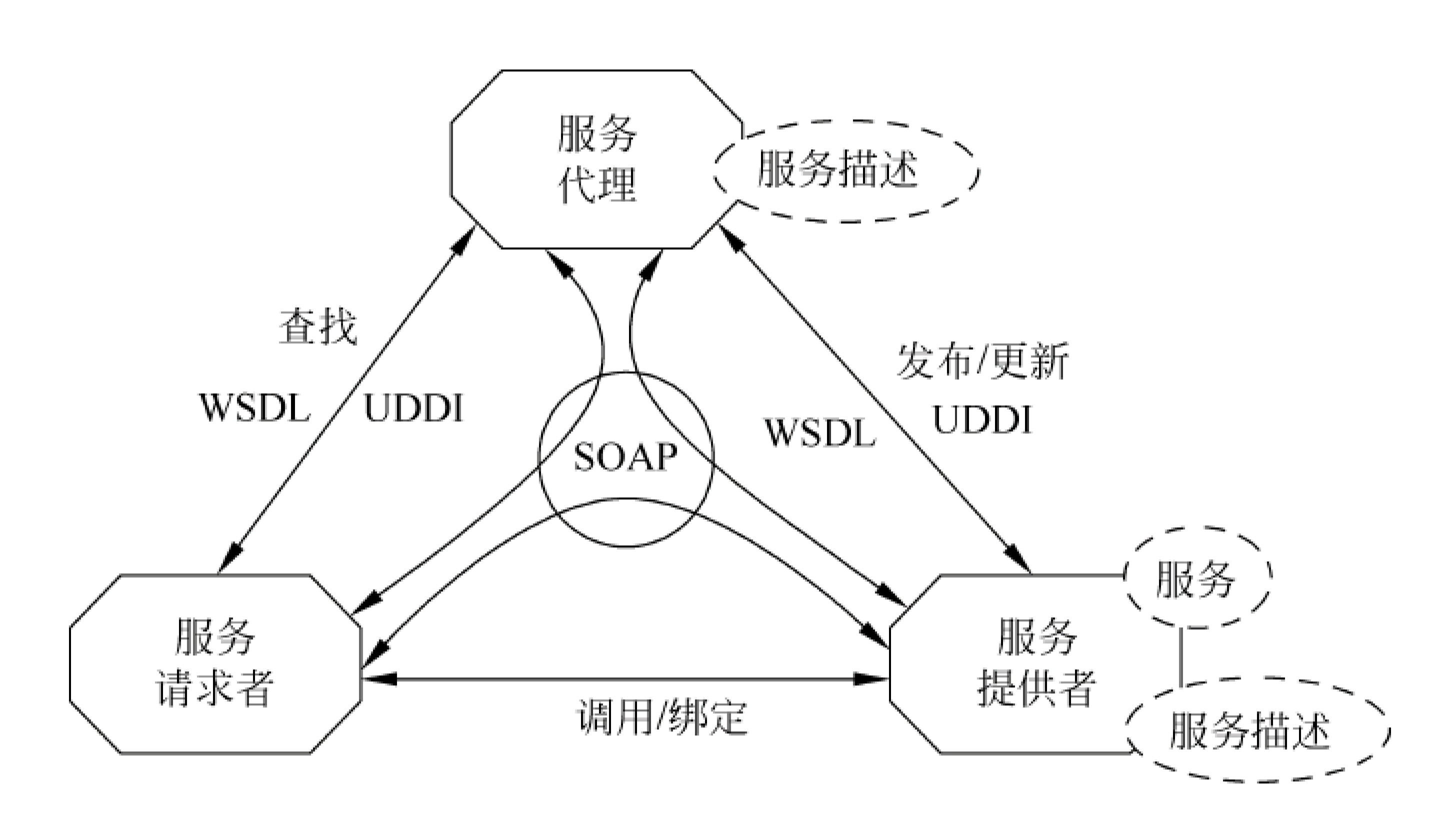


图 1-4 Web 服务体系结构

3. Web 服务相关技术标准

Web 服务相关技术标准如下。

(1) 可扩展标记语言(XML),它是 Web 服务的驱动力。XML 不是一种编程语言或者 API,而是一种独立于平台的组织数据的方式。XML 的语法便于通过编程来处理文本数据,同时又便于为人们所理解。Web 服务使用 XML 作为标准,在网络设备之间进行通信。

(2) 简单对象访问协议(Simple Object Access Protocol,SOAP)是在分散或分布式的环境中交换信息的简单的协议,开发人员可以使用这种独立于平台的机制,远程调用分布式对象的方法。SOAP 消息的通信使用 XML 来描述对象、方法以及执行的参数。客户机和服务器都可以实现和使用 SOAP。

相对于传统的公共对象请求代理体系结构(Common Object Request Broker Architecture,CORBA)和分布式组件对象模型(Component Object Model / Distributed Component Object Model,COM/DCOM)标准,SOAP 采用 HTTP 作为底层通信协议,RPC(Remote Procedure Call)作为一致性的调用途径,XML 作为数据传送的格式,允许服务提供者和服务客户经过防火墙在 Internet 进行通信交互,可以说 SOAP=HTTP+RPC+XML。

(3) Web 服务描述语言(Web Service Description Language,WSDL)是用 XML 文档来描述 Web 服务的标准,是 Web 服务的接口定义语言,它从句法层面对 Web 服务的功能进行描述,包括 Web 服务的三个基本属性:①服务做些什么,即服务所提供的操作方法;②如何访问服务,即和服务交互的数据格式以及必要协议;③服务位于何处,即协议相关的地址,如 URL。

WSDL 文档结构包括四个部分:Port Type(访问端口)、Binding(为每个端口定义消息格式和协议细节)、Message(Web 服务使用的消息)、Data Type(Web 服务使用的数据类型,使用 XML Schema 语法来定义数据类型)。WSDL 只是提供了 Web 服务的接口描述,对服务的行为约束和属性描述缺乏进一步的支持。

(4) 通用描述、发现和集成(UDDI)协议是为解决 Web 服务的发布和发现而制定的基于 Internet 的电子商务技术标准,它包含一组基于 Web 的、分布式的 Web 服务信息注册中

心的实现标准,以及一组使企业能将自己提供的 Web 服务注册到该中心的实现标准。UDDI 定义了一组基于 SOAP 消息的公用 SOAP API,用户利用 SOAP 消息来查找和注册 Web 服务,并为应用程序提供接口来访问注册中心。

服务注册(Service Registry)是一种服务代理,它是在 UDDI 上需要发现服务的请求者和发布服务的提供者之间的中介。当请求者决定使用特定的服务时,通常以借助于开发工具(如 Microsoft Visual Studio .NET)创建并发送服务请求,然后处理响应的方式访问服务的代码来绑定服务。

(5) 语义 Web 服务标记语言(OWL-S),是语义 Web 服务标记语言的标准,它比 WSDL 更能向用户提供可理解的服务资源的描述形式,提高服务选取与推荐的准确性。语义 Web 服务的主要方法是利用本体(Ontology)来描述 Web 服务,然后通过这些带有语义信息的描述实现 Web 服务来实现服务的自动发现、调用和组合。语义 Web 和 Web 服务是语义 Web 服务的两大支撑技术。OWL-S 是连接两大技术的桥梁,目前对语义 Web 服务标记语言研究最重要的组织就是 DARPA,其研究组 OWL Services Coalition 提出了语义 Web 服务标记语言(OWL-S)。

4. Web 服务技术的优势

Web 服务技术有如下优势。

(1) 平台无关、语言无关性。Web 服务技术的主要目标是在现有的各种异构平台的基础上构筑一个通用的平台无关、语言无关的技术层,各种不同平台之上的应用可以依靠这个技术层来实施彼此的连接和集成。

(2) 自描述能力。Web 服务的所有协议,包括 SOAP、WSDL、UDDI 都是 XML 文档,所以 Web 服务具有自描述的良好性质。

(3) 松耦合性。当一个 Web 服务的实现发生变更时,调用者是不会感到这一点的。对于调用者来说,只要 Web 服务的调用接口不变,Web 服务实现的任何变更对他们来说都是不透明的,甚至当 Web 服务的实现平台从 J2EE 迁移到 .NET 或者是相反的迁移流程,用户都一无所知。

(4) 易于集成。由于 Web 服务采用简单的、易理解的标准 Web 协议作为组件界面描述和协同描述规范,完全屏蔽了不同软件平台的差异,无论是 CORBA、DCOM 还是 EJB,都可以通过这一种标准的协议进行互操作,实现了在当前环境下最高的可集成性。

(5) 用消息传递代替传统的 APIs。Web 服务采用了 SOAP。SOAP 独立于平台,可以在不同的平台、环境下进行传递和交互。

Web Service 是组件技术在 Internet 中的延伸,从本质上讲是放置于网络上的可重用构件。从更高的概念层面讲,可以将 Web 服务视为一些工作单元,每个单元处理特定的功能任务。再往上一步,可以将这些任务组合成面向业务的任务,以处理特定的业务操作任务,从而使非技术人员去考虑一些应用程序,这些应用程序可以在 Web 服务应用程序工作流中一起处理业务问题。因此,一旦由技术人员设计并构建好 Web 服务之后,业务流程架构设计师可以聚集这些 Web 服务来解决业务层面的问题。

1.5 Web应用与发展趋势

Web,这个由无以计数的超链接形成的网络世界,在给每一个人的工作、学习、生活和娱乐带来无穷的便利和全新的生命体验的同时,新的技术、新的理念、新的应用不断出现,又让人们对 Web 的未来充满了无限的遐想。

1.5.1 浏览器/服务器计算模式

在 Web 出现以前,计算机的应用模式经历了单机应用到网络应用两个阶段,这些不同的计算模式有各自的优点和不足。Web 的出现使得一种围绕 Web 服务的计算模式成为当前计算机应用的主流模式,并推动了软件开发、软件应用、应用集成方式的重大改变。

1. 集中式计算模式

在计算机诞生和应用的初期,计算所需要的数据和程序都是集中在一台计算机上进行的,称为集中式计算。随着网络的发展,这种集中式计算往往形成一种由大型机和多个与之相连的终端组成的网络结构。当支持大量用户时,大型机自顶向下的维护和管理方式显示出集中式处理的优越性。它具有安全性好、可靠性高、计算能力和数据存储能力强以及系统维护和管理的费用较低等优点。但它也存在着一些明显的缺点,如大型机的初始投资较大、可移植性差、资源利用率低以及网络负载大等。

2. 客户机/服务器(C/S)计算模式

随着微型计算机和网络的发展,数据和应用逐渐转向了分布式,即数据和应用程序跨越多个节点机,形成了新的计算模式,这就是客户机/服务器(Client/Server,C/S)计算模式。C/S 模式是一种典型的两层计算模式,它将应用一分为二:前端是客户机,一般使用微型计算机,几乎所有的应用逻辑都在客户端进行和表达,客户机完成与用户的交互任务,具有稳健的数据操纵和事务处理能力;后端是服务器,可以使用各种类型的主机,服务器负责数据管理,提供数据库的查询和管理、大规模的计算等服务。

C/S 计算模式具有以下优点:通过异种平台集成,能够协调现有的各种 IT 基础结构;分布式管理;能充分发挥客户端 PC 的处理能力;安全、稳定、速度快;可脱机操作。但随着应用规模的日益扩大,应用程序的复杂程度不断提高,C/S 结构逐渐暴露出许多缺点,主要包括:它必须在客户端安装大量的应用程序(客户端软件),开发成本较高,移植困难,用户界面风格不统一,使用繁杂,不利于推广使用,维护复杂,升级麻烦,信息内容和形式单一,新技术不能轻易应用等。

3. 浏览器/服务器(B/S)计算模式

随着 Web 的出现,客户机/服务器模式表现出许多不足,特别是“胖”客户机和对局域网的依赖,已经不能适应 Web 的发展。人们需要利用互联网,将应用分布到整个 Web 中,而不是局限于企业局域网内部,这就导致了一种更加灵活的多级分布式计算模式,即浏览器/

服务器(Browser/Server,B/S)模式的产生和发展。

浏览器/服务器计算模式是一种基于 Web 的协同计算,是一种三层架构"瘦"客户机/服务器计算模式,概念模型如图 1-5 所示。

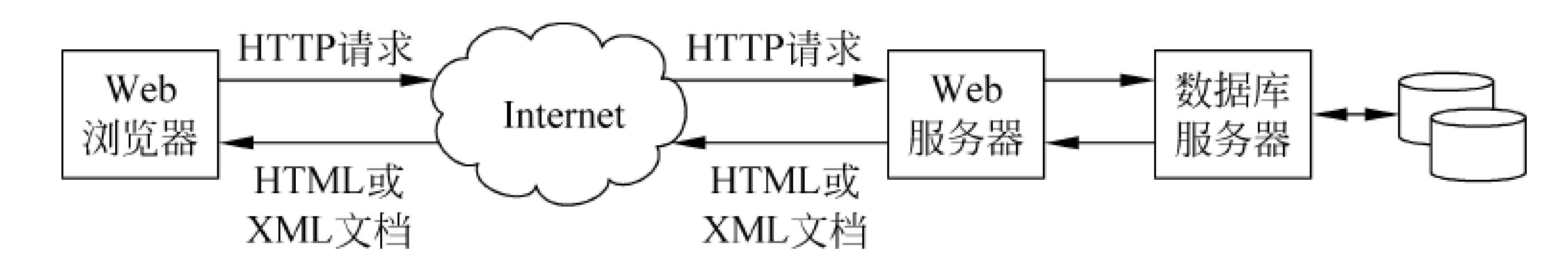


图 1-5 B/S 三层架构计算模式概念模型

第一层为客户端表示层,与 C/S 结构中的"胖"客户端不同,三层架构中的客户层只保留一个 Web 浏览器,不存放任何应用程序,其运行代码可以从位于第二层的 Web 服务器下载到本地的浏览器中执行,几乎不需要任何管理工作,是一种"瘦"客户机。第二层是应用服务器层,由一台或多台 Web 服务器组成,处理应用中的所有业务逻辑、对数据库的访问等工作。该层具有良好的可扩充性,程序的部署和管理主要在 Web 服务器上进行,相对于 C/S 而言无论是工作的复杂性还是工作量都大大减少。第三层是数据中心层,安装数据库服务器,负责整个应用中的数据管理。

B/S 计算模式与传统的 C/S 结构相比体现了集中式计算的优越性:具有良好的开放性,利用单一的访问点,用户可以在任何地点使用系统;用户可以跨平台以相同的浏览器界面访问系统;因为在客户端只需要安装浏览器,基本上取消了客户端的维护工作,有效地减少了整个系统的运行和维护成本。

目前,绝大多数的计算机应用都是部署在 Internet 上,例如各种各样的电子商务平台、企业和政府部门的业务系统等,大都使用了 B/S 结构。从应用的角度讲,用户使用系统,就是通过 Web 浏览器访问 Web 应用系统(网站)的过程。访问的基本流程如下。

(1) 在用户端,在浏览器地址栏中,用户输入要访问的网页网址 URL,按 Enter 键。

(2) Web 服务器收到用户的 HTTP 请求,根据 URL 中指定的路径和网页文件,调出相应的网页文件。如果用户要浏览的页面是普通的 html 页面,Web 服务器将把该页面直接发送给用户。如果是服务器页(如 jsp、asp 等),Web 服务器将把该页面交给应用服务器(如 Tomcat),由应用服务器执行页面中的服务器脚本程序,执行完后,将页面返给 Web 服务器,Web 服务器再将页面发送到用户端。

(3) 在用户端,Web 服务器返回的文档在浏览器中打开,即完成网页浏览。

1.5.2 SOA 软件设计模式

从计算机诞生到现在,计算机硬件技术在发展的同时,计算机软件也在悄悄地发生变化。这种变化不仅表现在计算机软件的内涵、应用方式上,同时也表现在软件的设计模式和开发方法上。软件开发思想的演化,使得在 Web 流行的今天,推动了面向服务的体系架构 SOA 的产生和发展,使其成为下一代软件体系架构的主流。

1. 软件体系架构设计与开发模式的演化过程

软件开发人员往往希望软件开发能够满足对于开发效率、可靠性、易维护性、易管理等

多方面的更高要求。无论在计算机发展的哪一个时期,这都是软件开发人员永恒的目标。可以把这种目标的实现分成以下几个阶段。

1) 模块化体系结构设计和结构化编程

在计算机诞生的初期,程序规模很小,基本上谈不上程序系统机构。直到20世纪70、80年代,随着软件规模的不断扩大,软件工程的概念开始出现。为了应对软件系统规模的不断扩大,人们开始考虑软件系统的体系结构问题,结构化编程思想开始兴起,出现了子程序、函数的概念。此时,各种各样的开发环境也不断出现,软件开发进入一种标准函数、用户自定义函数,实现了一种代码级的重用,有效地提高了软件系统的开发效率,提高了代码质量。在这一时期,软件开发模式、软件工程的思想日益融入到软件设计和开发人员中,出现了软件的生命周期开发模式、原型法开发模式等不同的软件开发模式。

生命周期开发模式又称软件开发瀑布模型,是面向功能或过程的软件开发方法。它将软件开发分成几个阶段:①用户需求分析,开发人员和业务人员交流,制定用户需求说明书;②系统设计,开发人员根据需求说明书进行系统设计,制定系统设计报告;③系统开发,根据系统设计报告,进行系统编码;④系统测试,系统实现后双方组织人员进行测试;⑤运行与维护,系统测试结束后,便进入系统的运行、维护期。

从理论上讲,软件开发的生命周期模式是非常科学的,但是利用生命周期模式开发系统基于两个假设:①用户能够清楚地、完整地提供系统要求;②开发者能完整地、严格地理解和定义要求。但在实际开发中,这两个假设是很难满足的。因为,首先,在开发初期,用户很难清楚地描述系统需求,或者系统需求将来可能发生较大变化;其次,开发人员和业务人员在交流时可能存在理解上的不一致,其结果是系统开发完毕后,不能很好地满足用户需求。生命周期模式是封闭式的,缺少灵活性,特别是在用户需求定义方面。

面向过程的生命周期法主要流行于20世纪80年代,直到现在,这种思想一直还对软件设计和开发人员有着很深的影响。

和生命周期模式不同,原型法开发模式要求经过对用户需求的简单快速分析,利用高级开发工具及环境,快速完成原型系统的设计和实施,提供给用户评价。一个原型系统就是系统的一个可运行的早期版本,它反映了最终系统的部分重要特征,在评价过程中,开发人员不断从用户那里得到反馈信息,修正原型的用户需求定义,进而对原型系统作相应改进,逐步减少分析与交互过程中的误解,弥补遗漏,从而提高最终系统的质量。

原型法的核心是用交互、快速建立起来的原型取代形式、不易修改的大块的规格说明,用户通过在计算机上实际运行和试用原型而向开发者提供真实的反馈意见。原型法开发模式的实现得益于面向对象的语言(Smalltalk、C++、Java等)和可视化的第四代开发工具的出现。从宏观上讲,原型法比生命周期法更实用,但是,在每一个原型的设计和开发过程中,都离不开生命周期的科学思想。

在软件工程的实践中,生命周期法和原型法的有效结合表现出了强大的生命力和可操作性。这种结合就是整个软件的开发表现为一个个原型的向前推进,在每一个原型的内部,又是按照生命周期的思想来设计。

2) 基于组件的软件体系架构与面向对象(组件)开发模式

进入20世纪90年代,随着网络的发展和软件开发规模的扩大,在涉及分布式、异构等复杂特征的环境中,代码级别的重用性差、可维护性差、效率低的弱点是不可逾越的,因此人

们以架构运行环境(如.Net、J2EE 等)来提供完善的支撑平台,从而把开发者解放出来,使其更专注于业务核心的开发。而这些业务功能以组件的形式(DCOM、EJB 等)发布运行在架构运行环境中,软件开发的重用模式也上升到业务组件的级别。

1991 年,对象管理组织(OMG)发布了公共对象请求代理体系结构(CORBA),它是一种标准的面向对象应用程序体系规范,由对象请求代理(Object Request Broker,ORB)、对象服务、公共设施、域接口和应用接口几个部分组成,核心部分是 ORB。ORB 提供了一种机制,通过这种机制,对象可以透明地发出请求和接收响应。分布的、可以互操作的对象可以利用 ORB 构造可以互操作的应用。ORB 可看做在对象之间建立客户/服务关系的一种中间件。基于 ORB,客户可以透明地调用服务对象提供的方法,该服务对象可以与客户机运行在同一台机器上,也可以运行在其他机器上通过网络与客户进行交互。ORB 截取客户发送的请求,并负责在该软件总线上找到实现该请求的服务对象,然后完成参数、方法调用,并返回最终结果。

CORBA 可以看做一种软件总线,它希望一个用户的软件系统是由运行在 CORBA 之上的一个个独立的软件组件对象构成。由于缺少企业协议的支持,或者说协议的研发未能跟上,CORBA 的思想实现得并不理想,随着 Web 服务的出现,CORBA 结构逐渐淡出。

1993 年,为解决代码的共享问题,微软公司提出了组件对象模型(Component Object Model,COM)。对于二进制代码的重用,最早是动态链接库(Dynamic Link Library,DLL),就为了动态调用功能而提出的,是一些函数的堆砌。COM 是一种组件技术,拥有状态、事件的概念,多个方法可以配合工作。COM 组件是遵循 COM 规范编写,以 Win32 动态链接库(DLL)或可执行文件(EXE)形式发布的可执行二进制代码,能够满足对组件架构的所有需求。遵循 COM 的规范标准,组件与应用、组件与组件之间可以互操作,可极其方便地建立可伸缩的应用系统。COM 是一种技术标准,其商业品牌称为 ActiveX。

1996 年,为了解决分布应用问题,微软提出了分布式组件对象模型(Distributed Component Object Model,DCOM),它扩展了 COM 的功能,使其支持不同计算机上对象之间的通信。DCOM 处理网络协议的低层次的细节问题,而不必关心太多的网络协议细节,从而使用户能够集中精力解决问题。DCOM 位于应用程序的组件之间,将组件以不可见的方式胶合在一起组成具有完整功能的应用程序。

在微软的系统上,COM/DCOM 发挥了应有的作用,但是 Internet 是存在巨大异构的环境,进入 2000 年后,随着 SOA 架构的兴起,COM/DCOM 技术也进入尾声。

3) 面向服务体系架构(SOA)与面向方面编程(AOP)

当软件的使用范围扩展到更广阔的范围,往往会面对更加复杂的 IT 环境和更加灵活多变的需求。服务(Service)的概念出现了,人们将应用(Application)以业务服务(Business Service)的形式公布出来供别人使用,而完全不需要去考虑这些业务服务运行在哪一个架构体系上,因为所有的服务都讲着同样的语言。此时 XML、Web Service 等技术的发展推动了 SOA 架构的形成和发展。

从技术的角度理解,面向服务体系架构(Service-Oriented Architecture,SOA)是构造分布式系统的方法,它将应用程序功能作为服务发送给最终用户或者其他服务。SOA 具有可重用、模块化和松耦合的特征。从业务的角度理解,将服务编排成各种业务系统,将业务逻辑用服务体现出来。SOA 体系架构可以应用在软件架构设计和应用集成两个不同的层面。

SOA同样也强调重用(Reuse),但是相对于传统的代码重用、对象重用和部件重用,SOA的重用粒度更粗。SOA的核心体现在企业应用或者业务功能上的“重用”和“互操作”,SOA的重用在于业务级的应用,即服务的重用,而不再把IT与业务对立起来,这可以被视为在IT驱动业务的方向上迈出的重要一步。

在SOA架构下,出现了面向方面编程(Aspect Oriented Programming,AOP)模式,AOP是OOP的延续,设计模式追求的是调用者和被调用者之间的解耦,AOP可以说也是这种目标的一种实现。AOP使原本复杂的调用与被调用和错综复杂的耦合关系变得清晰,使程序的整体架构保持高内聚、低耦合,这对于一个大型复杂系统来说是非常重要的。

2. SOA的设计思想

SOA是软件体系架构的下一代发展方向。SOA以可重用、模块化和松耦合为特征,将业务逻辑用服务体现出来。SOA体系架构可以应用在应用集成和软件架构设计两个不同的层面。

首先,从设计思路上看,SOA并不是一个新的概念,CORBA的出发点同样也是实现类似的目标,但是由于标准、中间件以及API的欠缺,使得CORBA技术只能纸上谈兵。今天,企业应用集成的迫切需求使得SOA再一次被提到前台,但是今天的SOA已经与以往CORBA时代所面临的环境大不相同了,其中最为核心的技术——Web服务已经拥有了一系列标准,WSDL、UDDI等标准都已经被融入到了各厂商的产品之中,为SOA的实现奠定了技术基础,使SOA技术的实现与应用成为可能,也成为SOA软件集成技术快速发展的动力,使SOA进入主流软件集成技术体系。

此外,SOA技术的逐步成熟,不仅将深刻改变软件集成的体系架构和工作方法,同时,也将彻底改变软件开发的思想、架构和工作方法。

SOA是一种思想、模式和体系,它规范了在软件架构以及系统集成中的方法,其思想的表现就是将业务逻辑和功能分解成更小的独立逻辑和功能单元。通过聚合技术,将这些单元构建成一个较大的业务逻辑单元,从而实现服务的独立存在,通过标准技术,使服务保持足够的共性,实现系统的体系化。

由IBM公司提案,国际开放群组(The Open Group)提出了一个SOA架构的参考模型,这个架构框架目前是产业界最权威和严谨的SOA架构标准。SOA标准模型的组成部分包括基础设施服务、企业服务总线、关键服务组件、开发工具和管理工具,如图1-6所示。

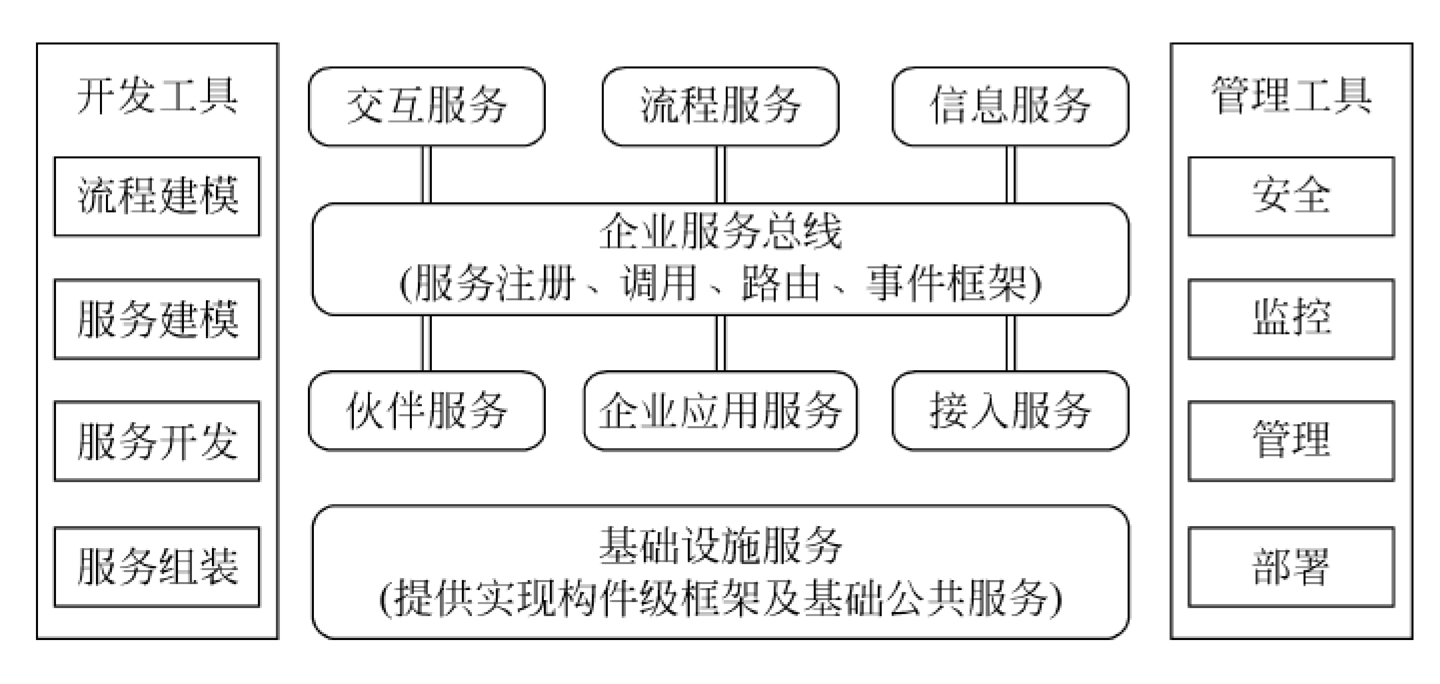


图1-6　SOA标准模型

上述 SOA 体系架构反映出基于 SOA 的软件设计模式和设计思想与传统的软件体系架构的不同。在该体系架构下，利用服务管理工具集，开发工具集，把最基本的共性业务抽象出来，封装为一个个基础服务，建立基础共性服务库。利用业务流程管理工具集，实现业务流程的定制和编排。完成后的服务在服务容器中通过服务总线进行协同。

3. SOA 的生命周期

由于 SOA 涉及业务的诸多方面，因此需要从一开始就对 SOA 项目进行细心的规划和设计。

(1) 建模(Model)。SOA 项目的第一步几乎和技术没有任何关系，所有事项都与用户的业务相关。面向服务的方法将业务所执行的活动视为服务，因此第一步是要确定这些业务活动或流程实际是什么。对用户的业务体系结构进行记录，这些记录不仅可以用于规划 SOA，还可以用于对实际业务流程进行优化。通过在编写代码前模拟或建模业务流程，可以更深入地了解这些流程，从而有利于构建帮助执行这些流程的软件。

建模业务流程的程度将依赖于预期实现的深度。对于企业架构师，需要对实际的业务服务进行建模。对于软件开发人员，可能对单个服务进行建模。

(2) 组装(Assemble)。对业务流程进行建模和优化后，开发人员可以开始构建新的服务和/或重用现有的服务，然后对其进行组装以形成组合应用程序，从而实现这些流程。在建模步骤中，已经确定了需要何种类型的服务以及它们将访问何种类型的数据。已经存在某种形式的实现这些服务或访问该类数据所需的一些软件。组装步骤将要找到已经存在的功能，并为其添加服务支持。另外，还涉及创建提供功能和访问数据源所需的新服务，以便满足用户的 SOA 涉及的业务流程范围内的需求。

(3) 部署(Deploy)。进行建模和组装后，要将组成 SOA 的资产部署到安全的集成环境中。此环境本身提供专门化的服务，用于集成业务中涉及的人员、流程和信息。这种级别的集成可帮助确保将公司的所有主要元素连接到一起协同工作。此外，部署工作还需要满足业务的性能和可用性需求，并提供足够的灵活性，以便吸纳新服务，而不会对整个系统造成大的影响。

(4) 管理(Manage)。部署后，需要从 IT 和业务两个角度对系统进行管理和监视。在管理步骤中收集的信息用于帮助实时地了解业务流程，从而能更好地进行业务决策，并将信息反馈回生命周期，以进行持续的流程改进工作。在这期间，将需要处理服务质量、安全、一般系统管理之类的问题。

在本步骤中，将监视和优化系统，发现和纠正效率低下的情况和存在的问题。由于 SOA 是一个迭代过程，因此，在此步骤中，不仅要找出技术体系结构中有待改进之处，而且还要找出业务体系结构中有待改进之处。

完成此步骤后就要开始新的“建模”步骤了。在“管理”步骤中收集的数据将用于重复整个 SOA 生命周期，再次进行整个过程。

(5) 控制(Government & Processes)。SOA 是一种集中系统，其中可以包含来自组织不同部门的服务，甚至还能包含来自组织外的服务。如果没有恰当的控制，这种系统很容易失控。控制对所有生命周期阶段起到巩固、支撑作用，为整个 SOA 系统提供指导，并有助于了解系统全貌。它提供指导和控制，帮助服务提供者和使用者避免遇到意外情况。

4. 企业服务总线

企业服务总线(Enterprise Service Bus,ESB)是SOA基础架构的关键组件,是SOA架构的一个支柱技术。作为一种消息代理架构,它提供消息队列系统,使用诸如SOAP或JMS(Java Message Service)等标准技术来实现。有人把ESB描述成一种开放的、基于标准的消息机制,通过简单的标准适配器和接口,来完成粗粒度应用(比如服务)和其他组件之间的互操作。

在SOA体系架构中,企业服务总线(ESB)提供了服务管理的方法和在分布式异构环境中进行服务交互的功能,通过ESB,实现服务的部署、配置、注册、消息处理、消息路由、交互、事件侦听、执行、服务质量和服务级别管理等。ESB由中间件技术实现并作为支持SOA的一组基础架构,支持异构环境中的服务、消息以及基于事件的交互,并且具有适当的服务级别和可管理性。

5. SOA的特征

面向服务的体系架构SOA具有如下特征。

(1) 服务的封装(Encapsulation)。把服务封装成可以被不同业务流程重复使用的业务组件,它隐藏所有实现细节,不管服务内部如何修改,使用什么平台、什么语言,只要保持接口不变,就不会影响最终用户的使用。

(2) 服务的重用(Reuse)。一个服务是一个独立的实体,与底层实现和用户的需求完全无关,极大地方便了服务的重复使用,从而降低了开发成本。

(3) 服务的互操作(Interoperability)。互操作并不是一个新概念,在CORBA、DCOM、Web Service中就已经采用互操作技术了。在SOA中,通过服务之间既定的通信协议进行互操作。主要有同步和异步两种通信机制。SOA提供服务的互操作特性更利于其在多个场合被重用。

(4) 服务是自治的(Autonomous)功能实体。服务是由组件组成的组合模块,是自包含和模块化的。SOA非常强调架构中提供服务的功能实体的完全独立自主的能力。传统的组件技术,如.NET Remoting、EJB、COM或者CORBA,都需要有一个宿主(Host或者Server)来存放和管理这些功能实体;当这些宿主运行结束时这些组件的寿命也随之结束。这样当宿主本身或者其他功能部分出现问题的时候,在该宿主上运行的其他应用服务就会受到影响。

在SOA架构中,非常强调实体自我管理和恢复能力。常见的用来进行自我恢复的技术,比如事务(Transaction)处理、消息队列(Message Queue)、冗余部署(Redundant Deployment)和集群(Cluster)系统在SOA中都起到至关重要的作用。

(5) 服务之间的松耦合度(Loosly Coupled)。服务请求者到服务提供者的绑定与服务之间应该是松耦合的。这就意味着,服务请求者不知道服务提供者实现的技术细节,比如程序设计语言、部署平台等。服务请求者往往通过消息调用操作,请求消息和响应,而不是通过使用API和文件格式。这个松耦合使会话一端的软件可以在不影响另一端的情况下发生改变,前提是消息模式保持不变。

(6) 服务是位置透明的(Location Transparency)。服务是针对业务需求设计的。需要

反映需求的变化,即所谓敏捷(Agility)设计。要想真正实现业务与服务的分离。就必须使服务的设计和部署对用户来说是完全透明的。也就是说,用户完全不必知道响应自己需求的服务的位置,甚至不必知道具体是哪个服务参与了响应。

SOA 的灵活性将给企业带来巨大的好处。如果把企业的 IT 架构抽象出来,将其功能以粗粒度的服务形式表示出来,每种服务都清晰地表示其业务价值,那么,这些服务的顾客就可以得到这些服务,而不必考虑其后台实现的具体技术。

基于以上特征,只要将 Web Service 置入特定的企业服务总线,就可以任意使用它。

6. SOA 和 Web 服务的关系

Web 服务与 SOA 有着很多相同的技术特点,如基于 XML,符合 SOAP、WSDL 和 UDDI 标准等。但是,SOA 不同于 Web 服务。SOA 是一种设计原则,是一个概念,是软件架构的方法学;Web 服务则属于技术规范,是一种具体的实现技术。Web 服务可以用来实现 SOA,但是如果没有 Web 服务,企业照样也可以很好地实现 SOA。

1.5.3 语义 Web

万维网已经成为人类最大的信息仓库,而且各种语言、各个知识领域的内容还在源源不断地快速增长。这些海量的信息在给我们提供便利的同时,让我们的信息查询变得极为困难。大多数的搜索引擎是基于关键词的搜索,搜索是基于页面内容的,若不是基于页面内容或页面信息的语义,则查准率较低。人们需要让页面内容有意义,从而提供各种依靠语义的自动化服务,这就是语义 Web 的研究动机。

1. 语义 Web 的概念

在 WWW 出现不久,人们就已经意识到语义对于 Web 的重要性。HTML 只是规范了信息的显示,却无法表达内容的含义。没有形式化的网页内容,机器将无法实现信息处理的自动化。只有将网页内容表述成机器可以理解的格式,Web 才可能成为一个巨大的知识库,充分实现信息的查找、共享和重用。为此,1998 年,万维网的发明者蒂姆·伯纳斯-李首次提出了语义 Web(Web Semantic)的概念。

对于语义 Web 的概念,一般的表述是:语义 Web 是当前 Web 的一个扩展,其中信息具有形式化定义的语义,更有助于计算机之间以及计算机与人之间的协同工作。其思想是使 Web 上的数据以这样一种方式来定义与链接,使其能够在各种不同的应用场景中有效地实现数据的发现、自动化处理、集成与复用。当且仅当 Web 不仅成为人所共享加工的场所,也成为自动化工具所共享加工的场所时,语义 Web 才能实现其全部潜力。

语义 Web 有很多突出的优点,包括数据集成更简单、搜索更精确、知识管理更方便等,语义 Web 的含义越来越丰富。

2. 语义 Web 的体系架构

要实现语义 Web,依赖于三大关键技术:XML、RDF 和 Ontology。语义 Web 分层体系架构如图 1-7 所示。

(1) Unicode 和 URI 层。Web 内容采用 Unicode 字符集,负责资源的编码,统一资源定

位符(Uniform Resource Identifier,URI)用于资源标识,唯一标识网络上的一个概念或资源。在语义 Web 体系结构中,该层是整个语义 Web 的基础。

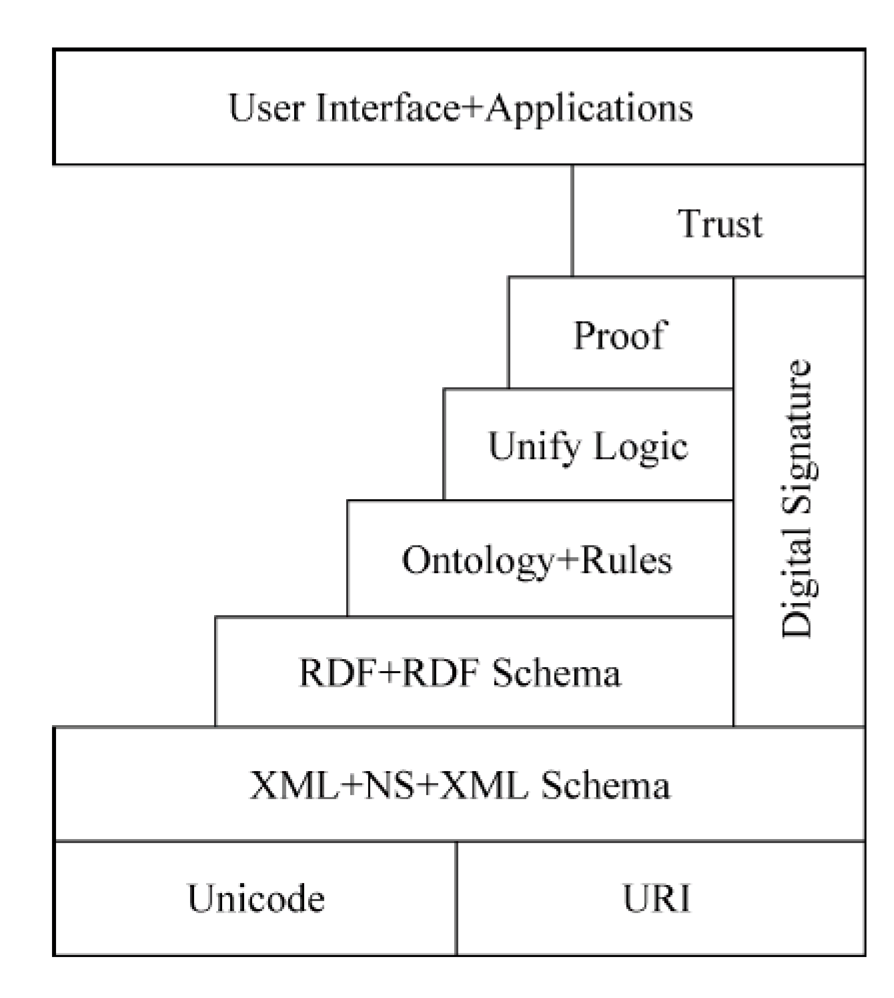


图 1-7 语义 Web 体系架构

(2) XML+NS+XML Schema 层。该层可以称为 XML 层,用于表示数据的内容和结构。XML 实现文档的结构化定义,即进行文档形式化。命名空间(Name Space,NS),由 URI 索引确定,目的是保证元素重明名而引起的歧义。XML Schema 提供更多的数据类型,为 XML 文档提供数据校验机制。该层负责从语法上表示数据的内容和结构,实现 Web 内容和表现形式的分离。

(3) RDF+RDF Schema 层。该层又可以分为 RDF 层和 RDF Schema 层,其中 RDF 用于描述资源及其相互关系;RDF Schema 层为 RDF 提供类型定义机制,确定 RDF 描述的资源所使用的领域词汇。因为,XML 不具备语义描述能力,W3C 推荐以 RDF(Resource Description Framework)标准来解决 XML 的语义局限。

RDF 解决的是如何采用 XML 标准语法无二义性地描述资源对象的问题,使得所描述的资源的元数据信息成为机器可理解的信息。RDF Schema 使用一种机器可以理解的体系来定义描述资源的词汇,其目的是提供词汇嵌入的机制或框架,在该框架下多种词汇可以集成在一起实现对 Web 资源的描述。

(4) Ontology + Rules 层。本体(Ontology)负责在 RDF(S)基础上定义的概念及其关系的抽象描述,用于描述应用领域的知识,描述各类资源及资源之间的关系,实现对词汇表的扩展。在这一层,用户不仅可以定义概念而且可以定义概念之间的复杂关系。规则用于描述领域知识中的前提和结论。本体和规则共同构成领域知识层。

W3C 推荐使用 OWL(Web Ontology Language)作为 Web 本体描述语言,OWL 既提供了正式的语义,又提供了附加的词汇,比起 XML、RDF 和 RDF Schema,对 Web 内容实现了更好的机器互操作性。

(5) Unify Logic 层。该层负责在下面各层基础上,提供公理和推理规则,而 Logic 一旦建立,便可以通过逻辑推理对资源、资源之间的关系以及推理结果进行验证,证明其有效性。

(6) Proof 层。通过 Proof 交换以及数字签名,建立一定的信任关系,从而证明语义 Web 输出的可靠性以及其是否符合用户的要求。

(7) Trust 层。支持代理间通信的证据交换,在用户间建立信任关系。

(8) User Interface + Applications 层。应用层是构建在语义 Web 之上的各种应用。

总之,在语义 Web 体系架构中,下面两层是语义 Web 的基础设施;中间从元数据发展到本体描述语言及其统一的逻辑是语义 Web 的关键;证明和信任及各层次贯穿的数字签名技术是扩充,是对语义 Web 成功应用的要求与展望。

3. 语义 Web 的应用

在语义 Web 中,可以提供各种依靠语义的自动化服务,包括如下两项。①互联网信息

发布与搜索。通过对内容的标注与分析从而克服关键词查询的歧义性，可以大大提高查询精度。此外，基于语义 Web 的文档检索与知识管理也是当前研究的一个热点。②Web 问题解答。在用 Ontology 对信息源进行标注的基础上，进一步运用知识库来解答用户的提问。例如，Stanford 大学研制的 Triple 系统是一个基于逻辑程序设计的 RDF 查询系统，逻辑子句的问题求解能力使它能够解答较为复杂的问题。德国 Karlsruhe 大学等单位研制的 SEAL 是一个语义 Web 门户网站，它具有回答用 F-逻辑表示的查询的能力，F-逻辑使得 Ontology 中的概念与问题求解规则融合于一体。

语义 Web 的目标是改善当今的 Web，它的主要思想是使语义信息成为计算机可处理的对象。要将 Web 语义化是非常困难的，语义 Web 很难一下子获得巨大成功，但是，它会一点点地渗透到现有的 Web 中，让人们不知不觉地进入语义 Web 的时代。

本章小结

本章首先介绍了 Internet 的产生和发展，然后介绍了万维网的概念以及万维网和 Internet 的关系；讲解了 Web 服务器、Web 浏览器的概念和功能并详细讲解了 HTTP 的原理。在此基础上，介绍了 WWW 中的常见概念，较详细地介绍了 Web 相关技术，包括 Java 技术、XML 技术和 Web 服务。从应用的角度出发，介绍了计算机应用模式的变迁；分别对集中式计算、C/S 计算模式和基于 Web 的 B/S 三层体系结构进行了介绍，并分析了 B/S 架构 Web 应用的基本工作过程。从软件开发的角度，介绍了软件系统体系架构及开发模式的发展，并详细讲解了 SOA 架构的设计思想、基本框架和应用模式。最后，简要介绍了语义 Web 的概念和发展。

习题 1

1. 上网查询以下 HTTP 头域的含义及取值：

Host、User-Agent、Accept、Accept-Charset、Keep-Alive、Connection、Date、Expired 、Referer、Server、Last-Modified、Etag、Via。

2. 什么是 Web 服务器和 Web 浏览器？简述它们的基本功能。

3. 什么是 B/S 结构？它和 C/S 结构相比有什么优点？

4. 画出 B/S 三层架构概念图，简述其基本工作机理。

5. 什么是 Web 应用？Web 应用和传统的 exe 程序有何不同？

6. Java 技术包括哪些方面的内容？

7. XML 技术标准体系是如何划分的？列举常用的 XML 基础标准，并说明它们的功能。

8. 什么是 Web 服务？Web 服务基于哪些主要的技术标准？

9. 什么是语义 Web？画出语义 Web 的分层模型，并说明各层的功能。

第2章 Web服务器的架设和管理

【本章导读】

在 Internet 中，Web 服务是最主要的网络服务之一，几乎一提到 Internet，就会想到万维网（World Wide Web，WWW）。实际上，WWW 只是 Internet 的一个子集，它是由 Internet 中的 Web 服务器和 Web 客户机构成的。Web 服务器就是那些安装了 Web 服务器软件的计算机，而安装了 Web 浏览器的计算机则是 Web 客户机。Web 技术不仅使 Internet 更加精彩，同时它还改变了计算机的应用模式，推动了基于 Web 的 B/S 三层架构的产生和发展，使传统的计算机应用软件发展到基于 Web 的应用，从各种各样的电子商务平台到企业和政府部门的应用系统，大都部署在 Web 上，Web 应用已经成为计算机应用的主要模式。

本章首先介绍 Web 服务器和操作系统的关系以及目前主要的 Web 服务器产品，然后以 Windows Server 中的 IIS 为例，介绍 Web 服务器的安装、配置和管理；讲解一个 Web 站点的主要概念，包括端口、主目录、默认文档、目录安全性等概念；介绍目前使用最多的 Apache 和 Tomcat 服务器的功能、两者的关系；介绍 Java 运行环境、JDK、JVM 的概念以及它们的安装和配置；讲解虚拟主机、虚拟目录等概念；最后，对 Web 服务器的远程管理与内容维护进行介绍。

【知识要点】

2.1 节：Web 服务器的概念、Windows Server/IIS 组件、Apache HTTP Server。

2.2 节：Internet 信息服务(IIS)组件、IIS 组件的安装、Internet 服务管理器。

2.3 节：新建 Web 站点、主机头、主目录、站点的启动、站点的停止、Web 应用目录结构的规划、文件命名、URL。

2.4 节：Web 站点属性、站点的配置、IP 地址及域名限制、匿名访问和验证控制、默认文档、HTTP 头。

2.5 节：Apache 服务器、Tomcat 应用服务器、Apache 服务器的安装、Apache 服务器的基本配置、Java 运行环境、Tomcat 的安装和配置、Apache 和 Tomcat 的整合。

2.6 节：虚拟主机的概念、基于 IP 地址的虚拟主机、基于域名的虚拟主机、Apache 中虚拟主机的配置、虚拟目录的概念、Apache 中虚拟目录的配置、Tomcat 中虚拟目录的配置。

2.7 节：远程管理、终端服务、远程桌面、FTP 服务。

2.1 操作系统与 Web 服务器

在全球数亿个网站或 Web 应用系统的背后，都运行着一个 Web 服务器。要使一台计算机成为 Web 服务器，需要安装网络操作系统和相应的 Web 服务组件。不同的操作系统平台，安装的 Web 服务组件也不相同。Web 服务组件从本质上讲就是一个网站或 Web 应用的运行环境，因此，安装的 Web 服务组件不同，Web 应用的开发工具也不相同。

目前，主流的 Web 服务器软件主要有微软的 IIS 和开源软件 Apache Web 服务器。IIS 支持 ASP 和 .NET 编程，只能运行在 Windows 平台下，Apache 支持 PHP、CGI、JSP 编程且可运行于多种操作系统平台。虽然 Apache 是世界使用排名第一的 Web 服务器平台，但是，由于 Windows 的易用性，因此具有很大的装机数量，IIS 的应用也很多。

2.1.1 Web 服务器

在 Internet 中，Web 服务器有两个层面的含义，一是指安装了 Web 服务器的计算机，二是指 Web 服务器程序。所谓 Web 服务器程序，简单地讲就是一个服务程序，它仅仅需要监听合适的端口，建立连接，然后发送数据。服务器程序的开发总是和客户端软件的开发相辅相成的。

20 世纪 90 年代，Web 服务器、浏览器、HTML 和 HTTP 都得到了快速的发展，这标志着 WWW 的诞生。美国国家超级计算应用中心(NCSA)开发了 httpd 代码，实现了 Web 服务器的功能。同时，NCSA 开发了最早的浏览器，即 NCSA Mosaic 浏览器，是后来的 Netscape、微软 IE 以及众多网页浏览器的鼻祖。虽然 NCSA Web 服务器不再被维护和继续开发，但是仍然可以免费下载其源代码，它是著名的 Web 服务器 Apache 的前身。

随着 Internet 的发展，人们对 Web 服务器的功能提出了更多的需求。Web 服务成为很多商务应用都必须面对和采用的技术时，就出现了很多不同 Web 服务器以满足这些不同的需求。在电子商务应用环境中，可伸缩性、可靠性和高级动态功能都是 Web 服务器应该具有的关键因素。此外，易于配置和管理对于新手来说也同样重要。对于这些所有的特性来说并没有任何一个特定服务器能完全满足需求，但是如果对 Web 服务的需求有明确的认识，可以从不同的 Web 服务器中选择适合的服务器。

2.1.2 主流 Web 服务器简介

Web 服务器产品很多，同一种 Web 服务器往往也有不同的操作系统版本，下面对各种不同的 Web 服务器进行简要介绍。

1. Internet Information Server

Internet Information Server(IIS)是 Windows 自带的服务器组件。在 Windows 2000 Server 中，自带了 IIS 5.0，Windows Server 2003 中自带了 IIS 6.0。现在，微软已经全部采用 Windows Server 2003/IIS 6.0，而不再安装 Windows Server 2000/IIS 5.0 了。

在 IIS 中，包含了一系列的 ASP 对象，负责 ASP 页面中服务端脚本程序的解析工作，同

时,为用户开发基于 Web 的应用提供一个开发环境。IIS 6.0 支持 Web 应用的.NET 开发环境。

作为 Windows 平台的内置服务组件,其优点是容易安装和管理,但是最大的缺点是只能安装在 Windows 平台中,不能在其他的操作系统平台上安装。

在 Windows 服务器上,除了安装 IIS 外,还可以安装 Apache 和 Tomcat。其中,Tomcat 是 Serverlet/JSP 的容器,可以运行 JSP 脚本程序,开发基于 JSP 的 Web 应用系统。

2. Apache HTTP Server

Apache HTTP Server 是 Apache 软件基金会[①]开发的一个 Web 服务器产品,它是 Apache 组织的开发爱好者们在美国伊利诺伊斯大学超级计算机应用程序国家中心(National Center for Supercomputing Applications,NCSA)开发的 NCSA HTTPd 服务器的基础上开发与维护的一个 HTTP 服务器,简称为 Apache 服务器。

在发展初期,Apache 主要是一个基于 UNIX 系统的服务器,它的宗旨就是建成一个基于 UNIX 系统的、功能更强、效率更高并且速度更快的 WWW 服务器。目前,Apache 是使用最广的 Web 服务器,有多个操作系统平台版本,可以运行在几乎所有的 UNIX、Windows、Linux 等主流操作系统平台上,并且很多类型的 UNIX 操作系统都集成了 Apache。

因为 Apache 服务器具有简单、高效、性能稳定、安全、免费等特性,已经成为最为广泛的 Web 服务器。许多大型的网站,例如 Google、Yahoo、阿里巴巴、新浪、百度、网易、搜狐等都采用 Linux 或 FreeBSD 等操作系统平台,并配置 Apache 服务器,构建自己的 Web 服务器。在版本上,大多数公司应用 Apache 2.0 或 Apache 2.2.x。

3. 其他 Web 服务器

除了 Apache 和 IIS 外,还有以下一些不同特点的 Web 服务器。

(1) Zeus Webserver 服务器。Zeus 是一个商业化的 Web 服务器产品,在 SMP 环境下有优秀的可伸缩性,并实现了常见的特性集合,如访问控制、动态内容产生和安全等。具有健壮、集成有集群支持的容错和负载平衡等特色,是高端应用的很好的选择。

(2) iPlanet 服务器。iPlanet 是 Sun、Netscape 和 AOL 公司联合生产的 Web 服务器产品,iPlanet 和其他 Netscape 产品一样具有很高的性能,而且 iPlanet 具有 Sun 公司 Java 的特性。iPlanet 具有现今高性能 Web 服务器的特性,相对于其他 Web 服务器,iPlanet 能够提供更多的 Java 功能,能够运行标准 Java API,并且在 Java API 环境下运行速度良好。

(3) AOLserver Web 服务器。AOLserver 的研究始于 1994 年,当时它作为 Web 发布系统的一部分进行开发。在该 Web 发布系统中内嵌了 Web 服务器的 WYSIWYG(What You See Is What You Get,所见即所得)网页编辑器,该网页编辑器强调内容变化的便利性和内容更新的快捷性。早期的 Web 发布系统被设计成一个完整的网页编辑系统。随着时间的推移,AOL 公司的网页编辑器已经不复存在,但是由于 TCL 脚本语言的出现和它对动

---

① Apache 软件基金会(Apache Software Foundation,ASF),是专门为支持开源软件项目而办的一个非盈利性组织,正式创建于 1999 年,它的创建者是一个自称为“Apache 组织”(北美当地一支印第安部落的名称)的群体。ASF 支持了大量的 Apache 项目和子项目,例如 Apache HTTP Server、Tomcat、Struts 等。

态网页的支持，AOLserver 却生存了下来。

Web 服务器产品很多，随着互联网技术的发展，新的 Web 服务器产品还将不断出现。选择 Web 服务器时，服务器对动态脚本语言、API 的支持以及数据库连接的性能都是重要的因素。此外，Web 服务器的配置和管理，也是选择 Web 服务器的重要因素。

2.2 使用 Internet 信息服务

在 Windows 操作系统平台上，内置了 Internet 信息服务(Internet Information Server，IIS)组件，用于创建 Web 服务器。作为 Windows 平台，最常用的是 Windows 2000 Server 和 Windows Server 2003，其中 Windows 2000 Server 内置了 IIS 5.0，Windows Server 2003 内置了 IIS 6.0。目前，Windows Server 2003/IIS 6.0 是 Windows 平台最主要的 Web 服务器配置。

2.2.1 什么是 Internet 信息服务

IIS 是一组 Windows 操作系统组件，此组件可以使公司很方便地创建自己的 Web 服务器、FTP 服务器、E-mail 服务器和 NNTP 服务器，从而将信息和业务应用程序发布到 Web 中。在 IIS 内部，本质上是由一系列的 ASP 对象组成的，负责 ASP 页面中服务端脚本程序的解析工作，同时，为用户开发基于 Web 的应用提供一个开发环境。

在 Windows 2000 中，内置了 IIS 5.0。从 Windows Server 2003 开始，IIS 升级为 IIS 6.0，将 IIS 5.0 中的 SMTP 服务器升级为完整的 E-mail 服务器。IIS 由若干可选组件构成，用户可以根据需要选择不同的组件进行安装和配置，IIS 包含如下组件。

(1) Internet 服务管理器。用于配置和管理 IIS，可以在 MMC 中以管理单元形式显示，该管理工具还在控制面板的“管理工具”文件夹中创建一个快捷方式。

(2) Internet 服务管理器(HTML)。基于 HTML 的 Internet 服务管理器，可以使用浏览器对 IIS 进行远程管理。

(3) NNTP Service。NNTP(Network News Transfer Protocol，网络新闻传输协议)，是 TCP/IP 套件的成员。负责将新闻函件分发到 Internet 上的 NNTP 服务器和 NNTP 客户端，设置 NNTP 后，就可以将新闻文章存储在服务器上的中央数据库，用户可以选择指定的项目阅读。

(4) SMTP Service。SMTP(Simple Mail Transfer Protocol，简单邮件传输协议)，是 TCP/IP 套件的成员，用来管理邮件代理之间的电子邮件交换。

(5) World Wide Web 服务。Web 服务，用于对 Web 站点的创建、管理以及为用户访问 Web 服务器提供服务，内置服务端脚本引擎，是 ASP 等服务器脚本规范的容器。

(6) 文档传输协议(FTP)服务器。用于建立 FTP 站点，支持文件的上传和下载。配置 FTP 服务器，可以有效地对 WWW 服务器站点内容实行远程维护。

2.2.2 安装 IIS

安装 Windows 2000 Server 时，默认情况下，IIS 5.0 被一并安装。在 Windows Server 2003 中，IIS 组件是“应用服务器”的一部分，可以在安装操作系统时选择安装，也可以通过

“添加/删除 Windows 组件”方式来安装 IIS，或者通过管理工具中的“管理您的服务器”程序添加“应用程序服务器”角色，来完成 IIS 6.0 的安装。

下面以 Windows Server 2003 企业版为例，说明 IIS 的安装。具体操作步骤如下。

(1) 将 Windows Server 2003 系统光盘插入光盘驱动器。

(2) 在“控制面板”窗口中，双击“添加/删除程序”图标；在“添加/删除程序”窗口中，单击“添加/删除 Windows 组件”按钮，打开“Windows 组件向导”对话框；在组件列表中，选择“应用程序服务器”(在 Windows 2000 Server 中为 Internet 信息服务)，然后单击“详细信息”按钮，打开“应用程序服务器”对话框，如图 2-1 所示。

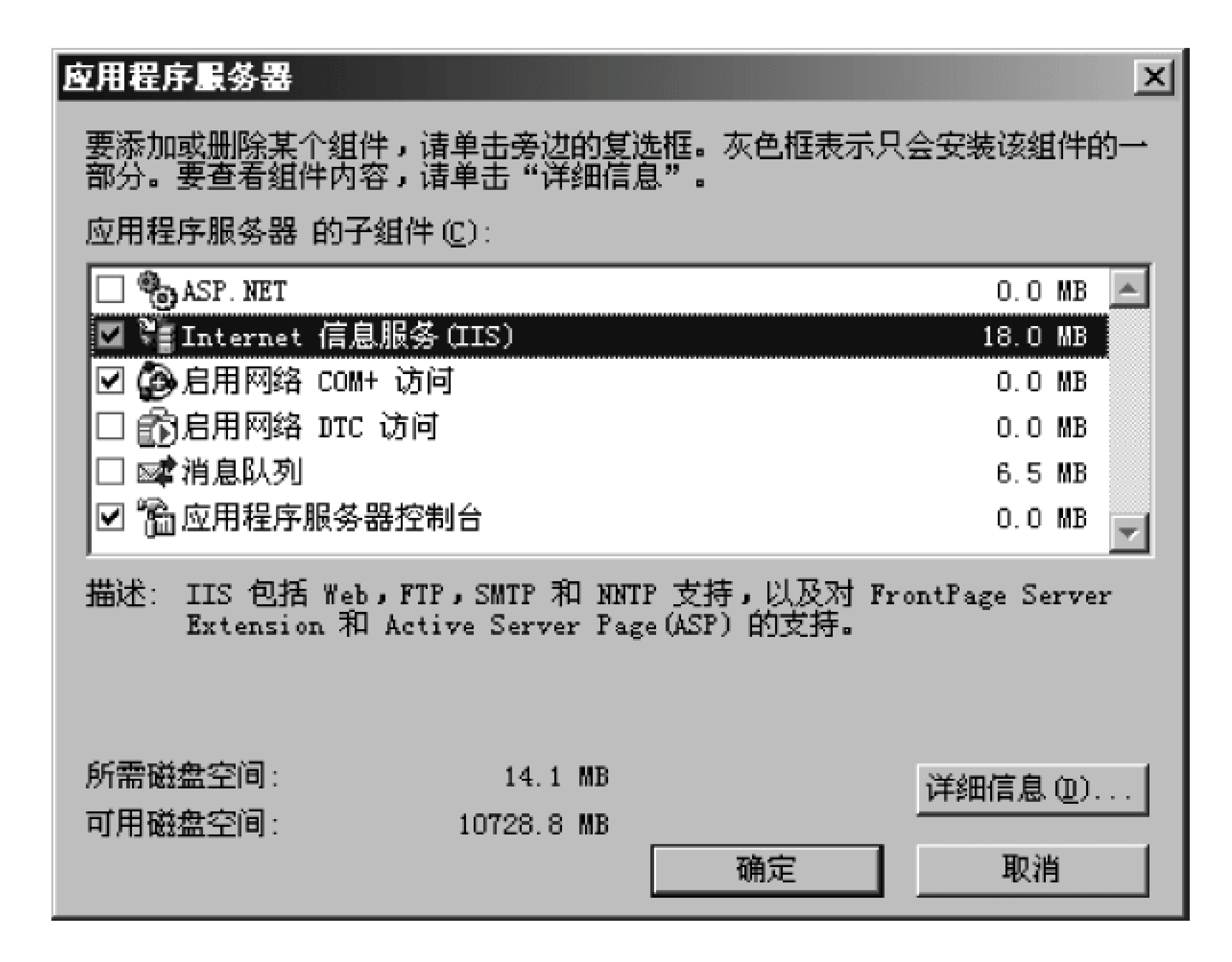


图 2-1 “应用程序服务器”对话框

(3) 在应用程序服务器组件列表中，选择“Internet 信息服务”，然后单击“详细信息”按钮，打开“Internet 信息服务(IIS)”对话框，显示 Windows Server 2003 中相关组件，如图 2-2 所示。

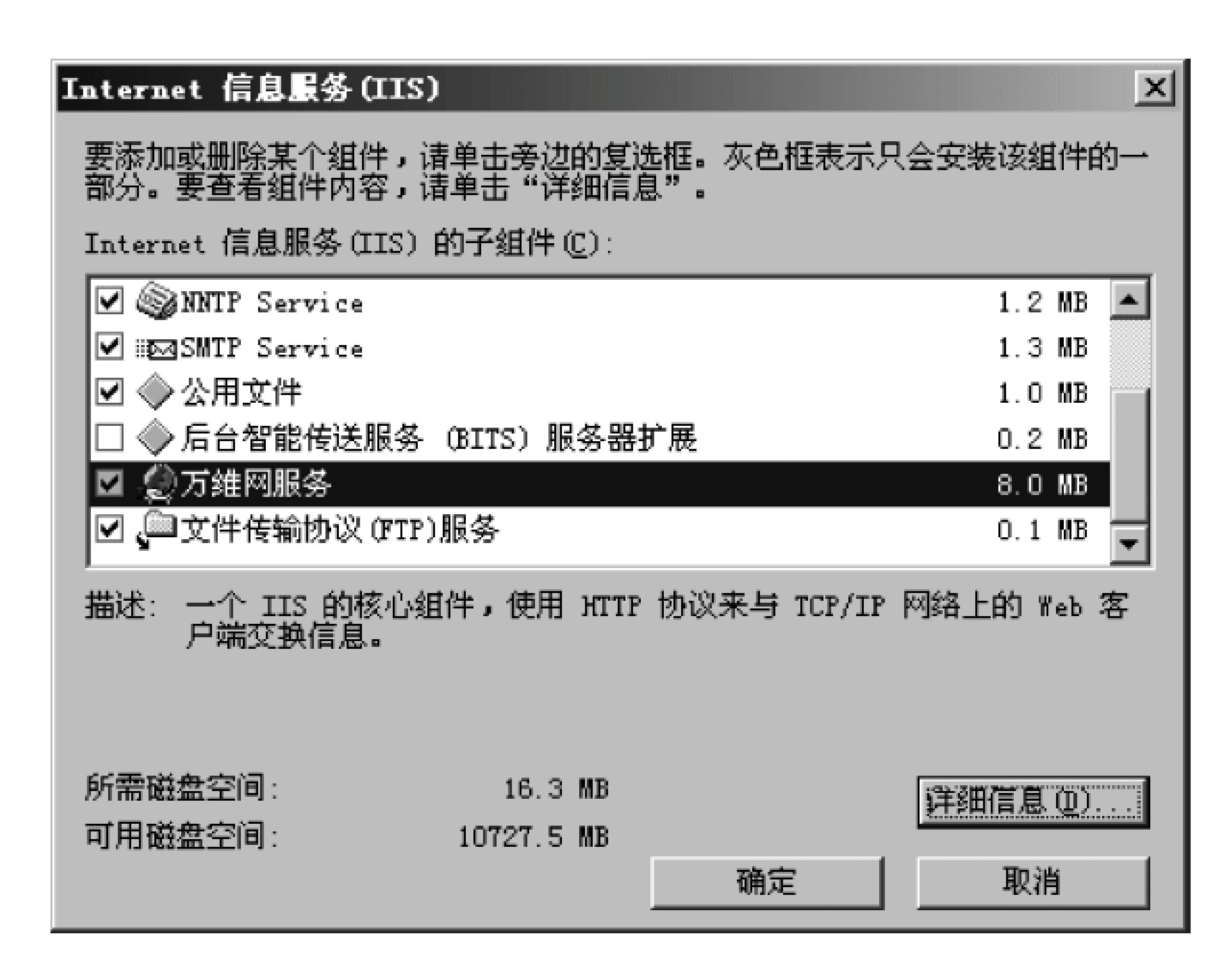


图 2-2 Internet 信息服务组件列表

(4) 选中“万维网服务”复选框，然后单击“详细信息”按钮，打开“万维网服务”对话框，显示万维网服务子组件列表，如图 2-3 所示。

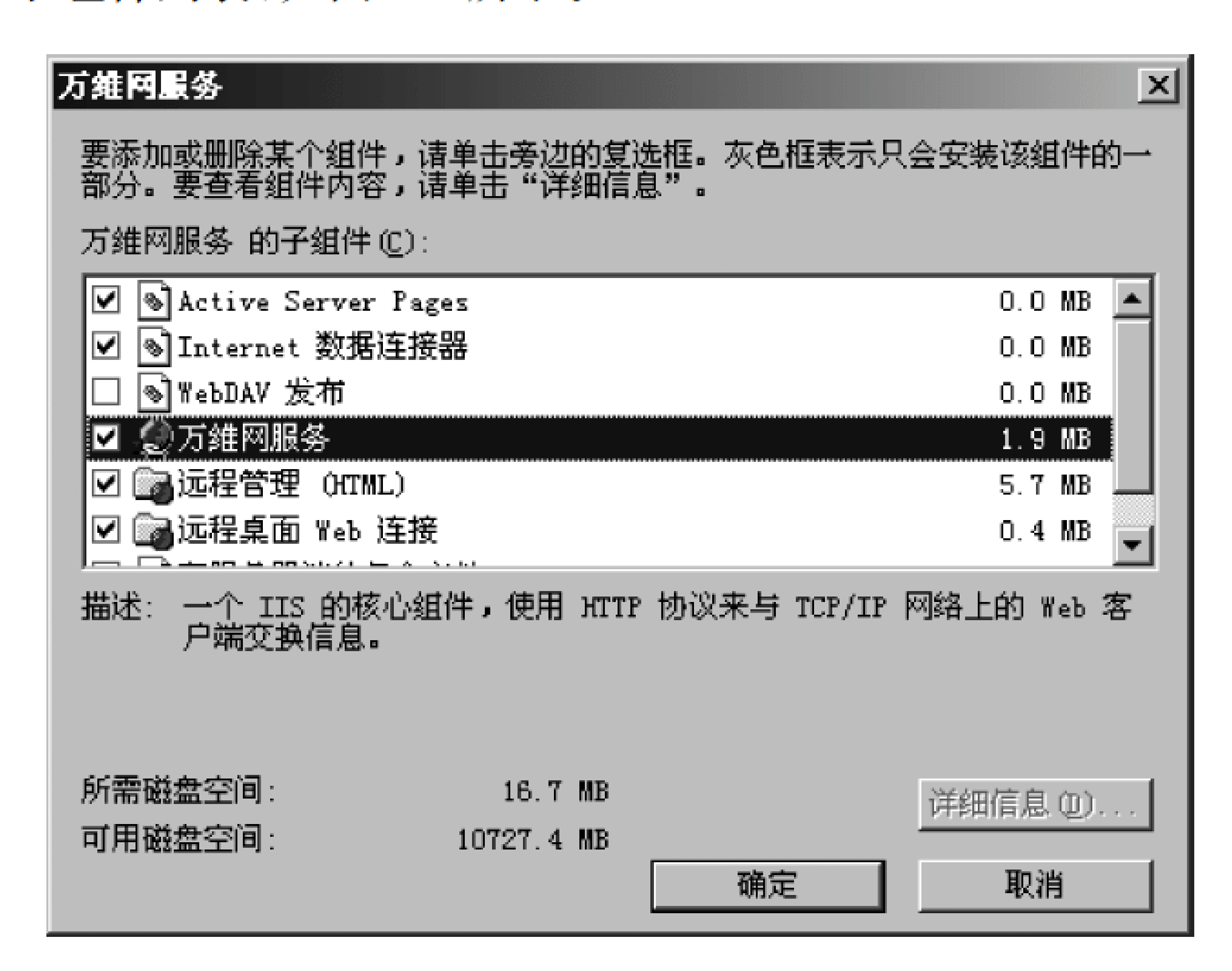


图 2-3 “万维网服务”对话框

(5) 在万维网服务子组件列表中，选择相应的组件，包括“万维网服务”、“远程管理(HTML)”和“远程桌面 Web 连接”，然后单击“确定”按钮，向导从光盘复制文件并进行相关的配置。安装结束后，在“控制面板”的“管理工具”中将增加“Internet 信息服务(IIS)管理器”、“远程桌面”等程序。同时，在服务器 C 盘根目录下将创建一个 Inetpub 文件夹，在该文件夹下创建多个子文件夹，文件夹结构如图 2-4 所示。



图 2-4 安装 IIS 6.0 后自动创建的相关文件夹

各文件夹说明如下。

① AdminScripts 文件夹：存储 CGI 脚本的根目录。

② ftproot 文件夹：ftp 服务根目录。

③ mailroot 文件夹：SMTP 服务器根目录。

④ nntpfile 文件夹：新闻组信息根目录。

⑤ wwwroot 文件夹：默认 Web 站点的根目录。

2.2.3 Internet 信息服务管理器概述

IIS 安装完成后，在 Web 服务器的"管理工具"文件夹中增加"Internet 服务管理器"工具。同时在"计算机管理"控制台中，在"服务和应用程序"节点下增加"Internet 信息服务"节点。通过 Internet 服务管理器，可以创建 Web 站点、FTP 站点，以及对它们进行配置和管理。对 IIS 的管理可以有多种途径，主要如下。

1. Internet 信息服务管理器

单击"开始"按钮，指向"程序"、"管理工具"，执行"Internet 信息服务(IIS)管理器"命令可以直接启动 Internet 信息服务管理器，如图 2-5 所示。



图 2-5 "Internet 信息服务(IIS)管理器"控制台

在安装 IIS 的 Web 服务器时，系统创建了两个 Web 站点，一个是"默认站点"，端口号为 80，另一个为 Administration，端口号为 8098。其中，默认网站主要用于初学者学习，Administration 站点则可以对 Web 服务器实施远程维护，即登录该站点即可。

2. Internet 信息服务管理单元

安装 IIS 后，Internet 服务管理器作为一个管理单元，被组织到"计算机管理(本地)"控制台中。在"控制面板"→"管理工具"中，双击"计算机管理"，打开"计算机管理"控制台，可以显示"Internet 信息服务(IIS)管理器"节点，如图 2-6 所示。

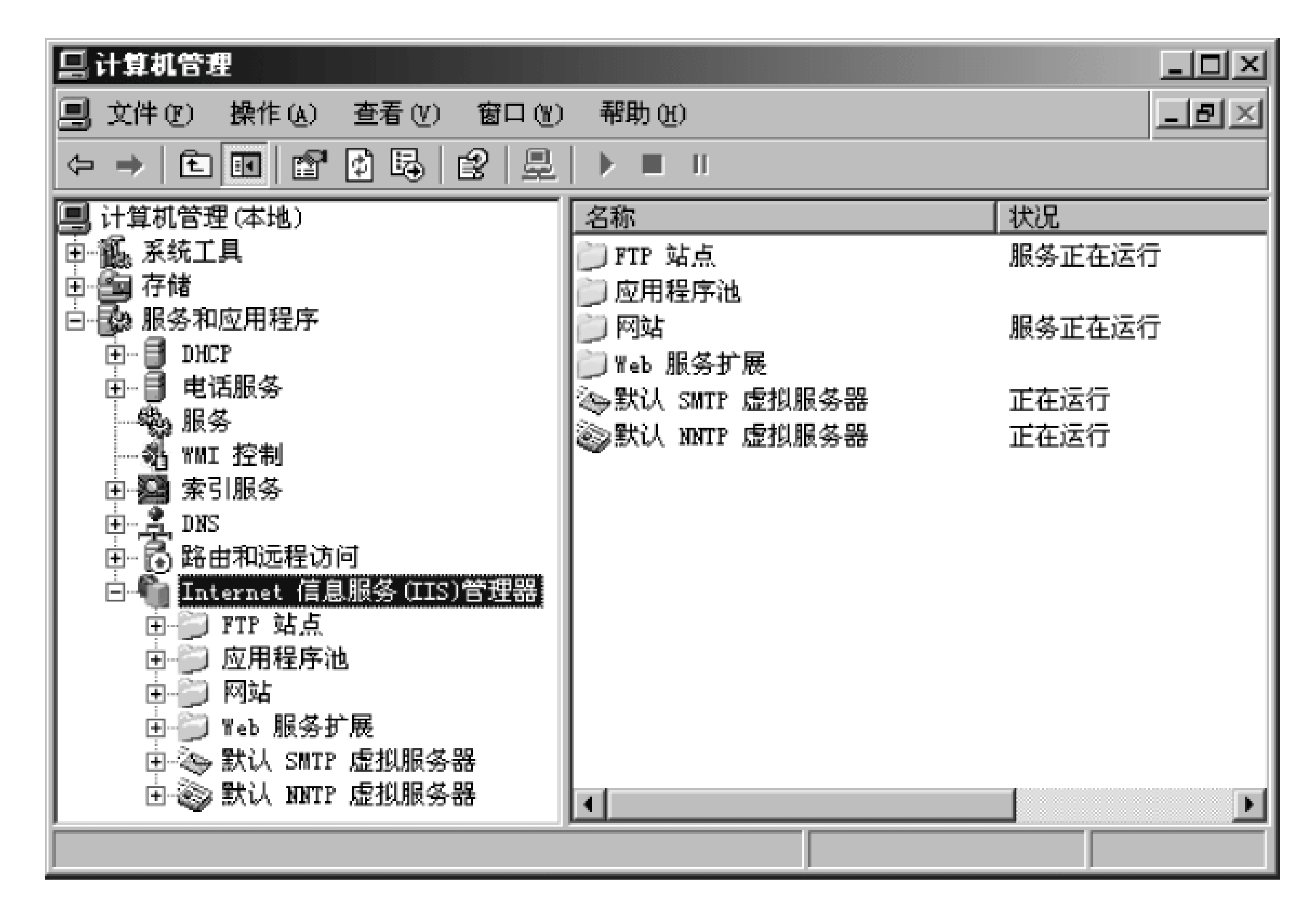


图 2-6　计算机管理控制台中的"Internet 信息服务(IIS)管理器"管理单元

2.3　Web 站点

在 Windows 平台中，当安装了 IIS 后，就可以创建 Web 站点了，从而把该计算机配置为 Internet 中的一台 Web 服务器，向用户提供 Web 连接服务。

2.3.1　创建 Web 站点

在 Windows 服务器/IIS 中，可以利用 IIS 创建和管理 Web 站点。下面以 Windows Server 2003 企业版为例，介绍 IIS 中 Web 站点的创建过程。

(1) 单击"开始"→"程序"→"管理工具"→"Internet 服务(IIS)管理器"，打开"Internet 信息服务"控制台，右击"网站"节点，打开快捷菜单，如图 2-7 所示。



图 2-7　打开快捷菜单

(2) 选择"新建"→"网站"命令,打开"网站创建向导";单击"下一步"按钮,打开"网站描述"界面,如图 2-8 所示。

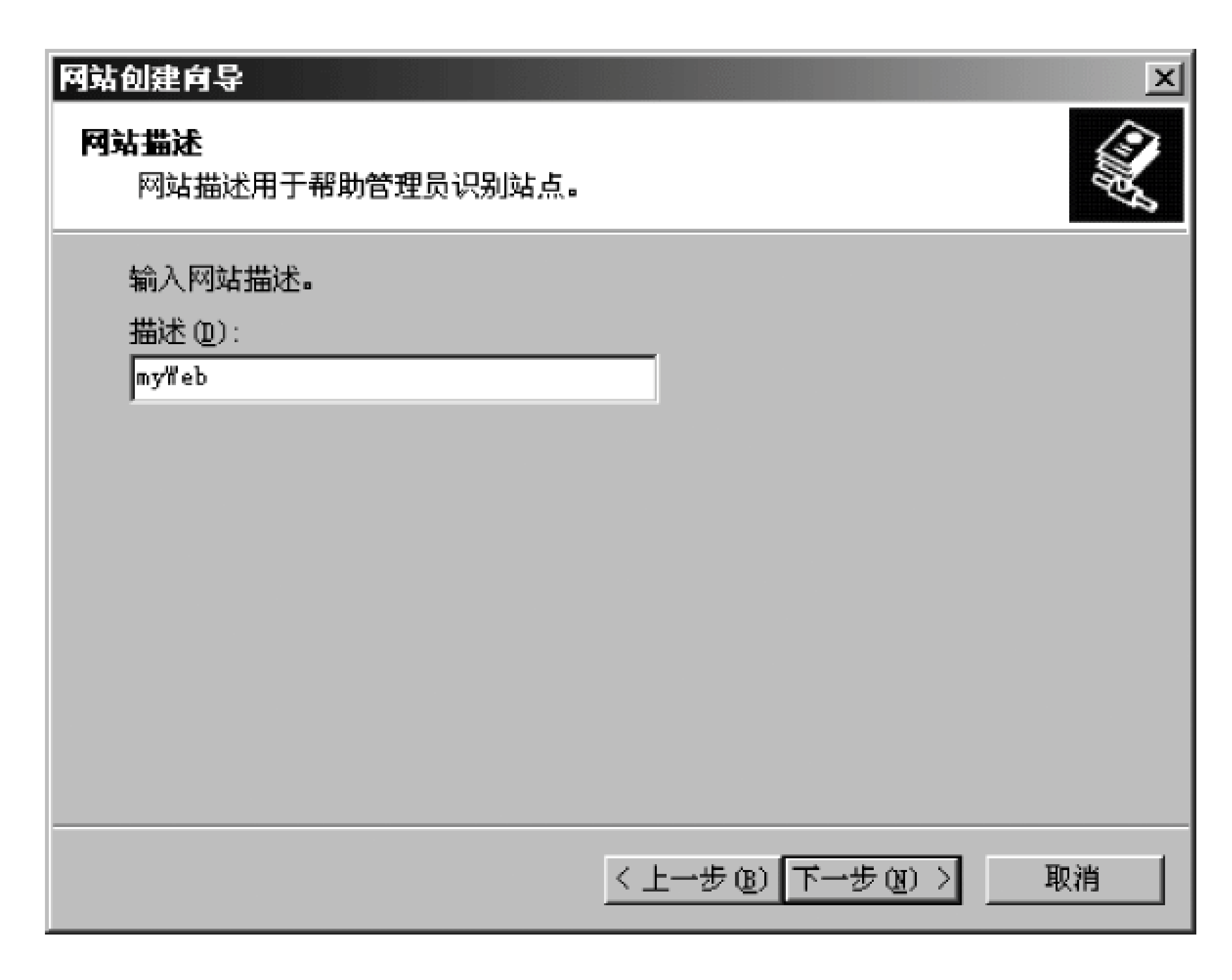


图 2-8 输入网站描述

(3) 在"网站描述"界面,输入 Web 站点的说明(即新站点的名称),该名称将在"Internet 信号服务(IIS)管理器"控制台中显示。单击"下一步"按钮,打开如图 2-9 所示的对话框。

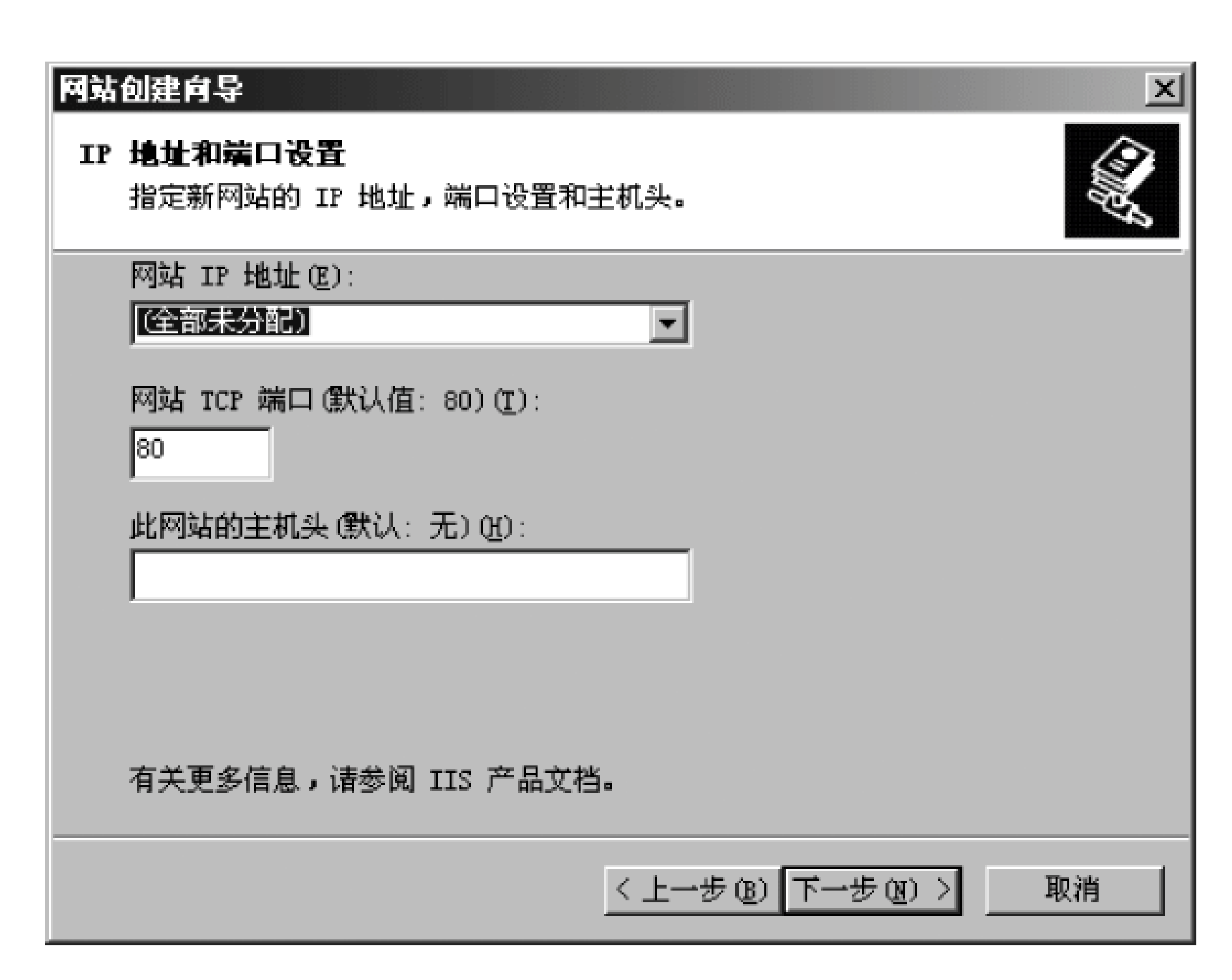


图 2-9 "IP 地址和端口设置"对话框

(4) 在"网站 IP 地址"下拉列表框中,默认显示"(全部未分配)"。单击下拉列表框,可以显示网卡设置的多个 IP 地址。可以选择"(全部未分配)"或从中选择一个 IP 地址。

(5) 根据开放系统互联模型(OSI 模型),所有的计算机通信程序都应有一个特定的端口号,它是通信程序的标识,用于发送和接收特定的数据包。因此,在服务器上,每一个服务程序也需要设置一个唯一的端口号。对于 Web 服务器,使用 HTTP 通信,默认的端口号为 80,如果运行多个 Web 站点,可以修改该端口号。此时,采用默认端口 80,该设置可以在以

后的网站管理中很容易地修改。指定 80 以外的端口号，要连接到该站点，在网址(IP 地址或域名)后，需要给定对应的端口号，如 http://202.194.73.118:8001/。

(6) 主机头即域名的概念，它也是在一台 Web 服务器上运行多个 Web 站点的方式之一。要运行多个 Web 站点，除了设置不同的端口号外，还可以注册多个域名，为每个站点设置不同的主机头，即域名。这里全部选用默认值，单击"下一步"，打开如图 2-10 所示的对话框。

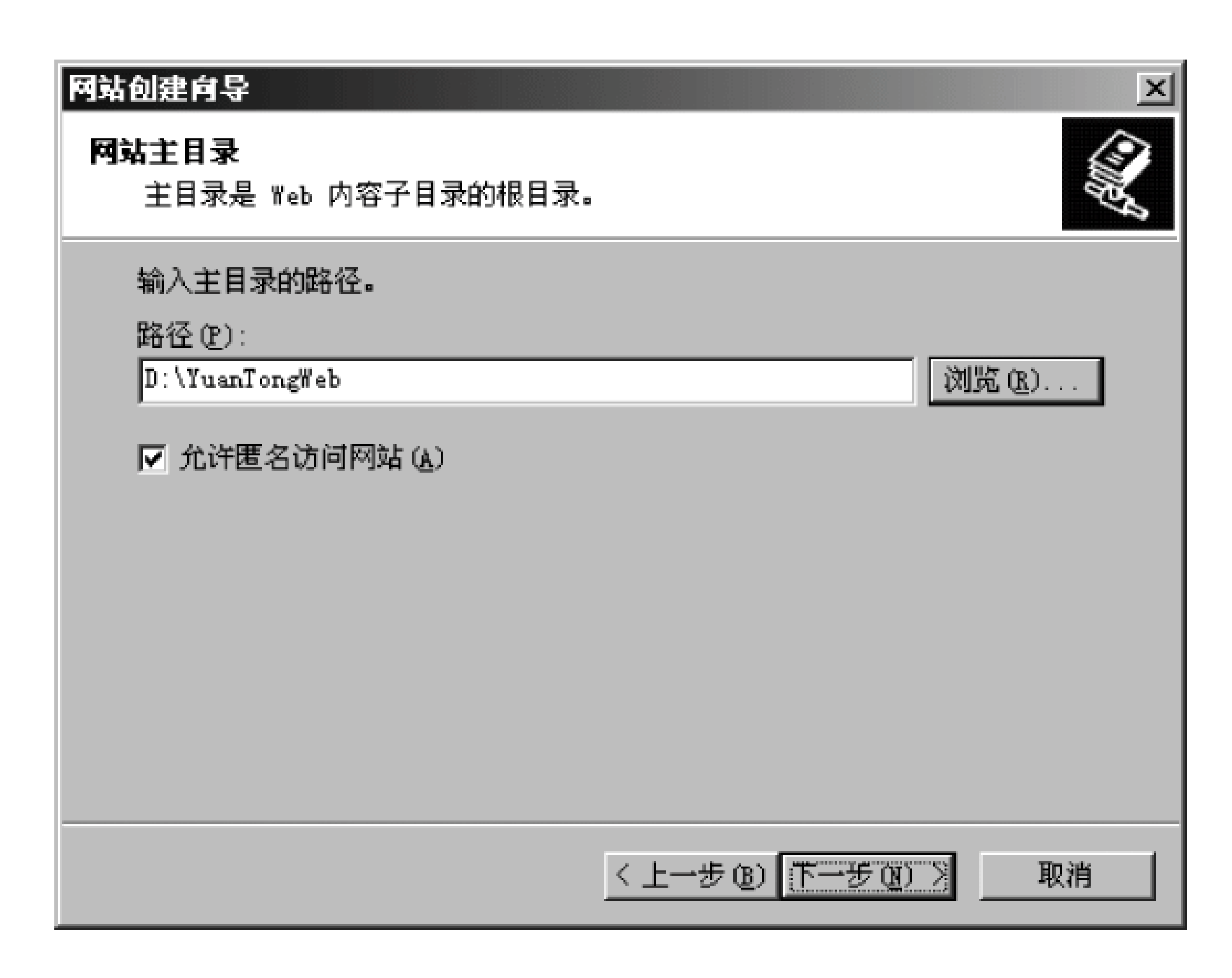


图 2-10　"网站主目录"对话框

(7) 从组成上来讲，所谓一个"网站"，是由一个文件夹及其包含的子文件夹和大量的网页文件、图片等资源文件构成的，这个文件夹称为站点的主目录。要设置网站主目录，单击"路径"文本框右侧的"浏览"按钮，选择一个目录作为网站主目录即可。主目录保存了一个 Web 站点中的所有内容，包括各个子文件夹以及所有的网页文件。站点主目录又称为站点的根目录，站点首页文件通常存储在站点的主目录下。

(8) 单击"下一步"按钮，打开"网站访问权限"界面，如图 2-11 所示。

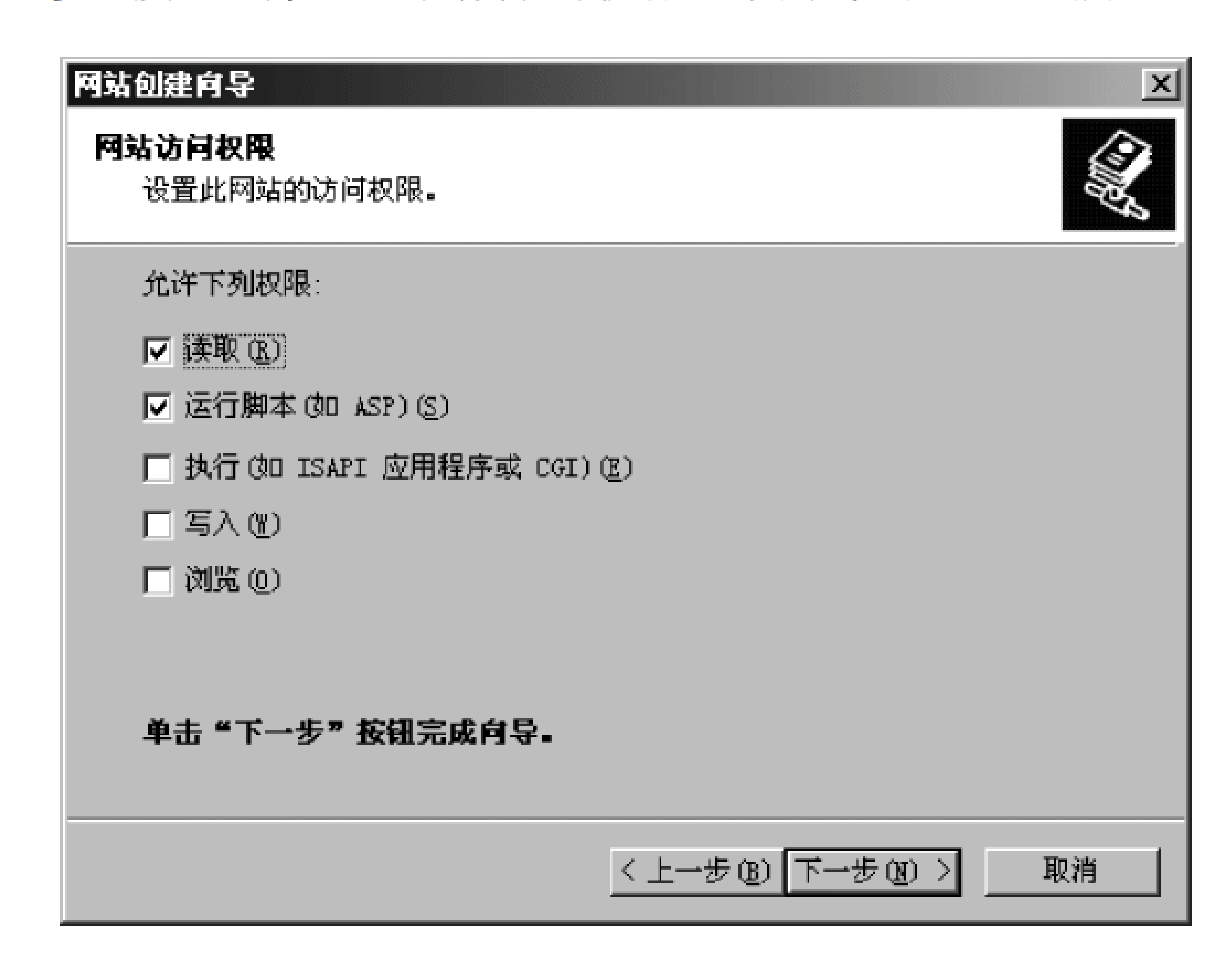


图 2-11　"网站访问权限"界面

(9) 根据需要,对 Web 站点的权限进行设置,从允许的权限中选择相应的权限。一般情况下,需要选择“读取”权限和“运行脚本(如 ASP)”权限。有关权限设置的详细介绍参见 2.4 节。

(10) 单击“下一步”按钮,打开“已经成功完成 Web 站点创建向导”界面。单击“完成”按钮,返回到“Internet 信息服务(IIS)管理器”控制台,如图 2-12 所示。

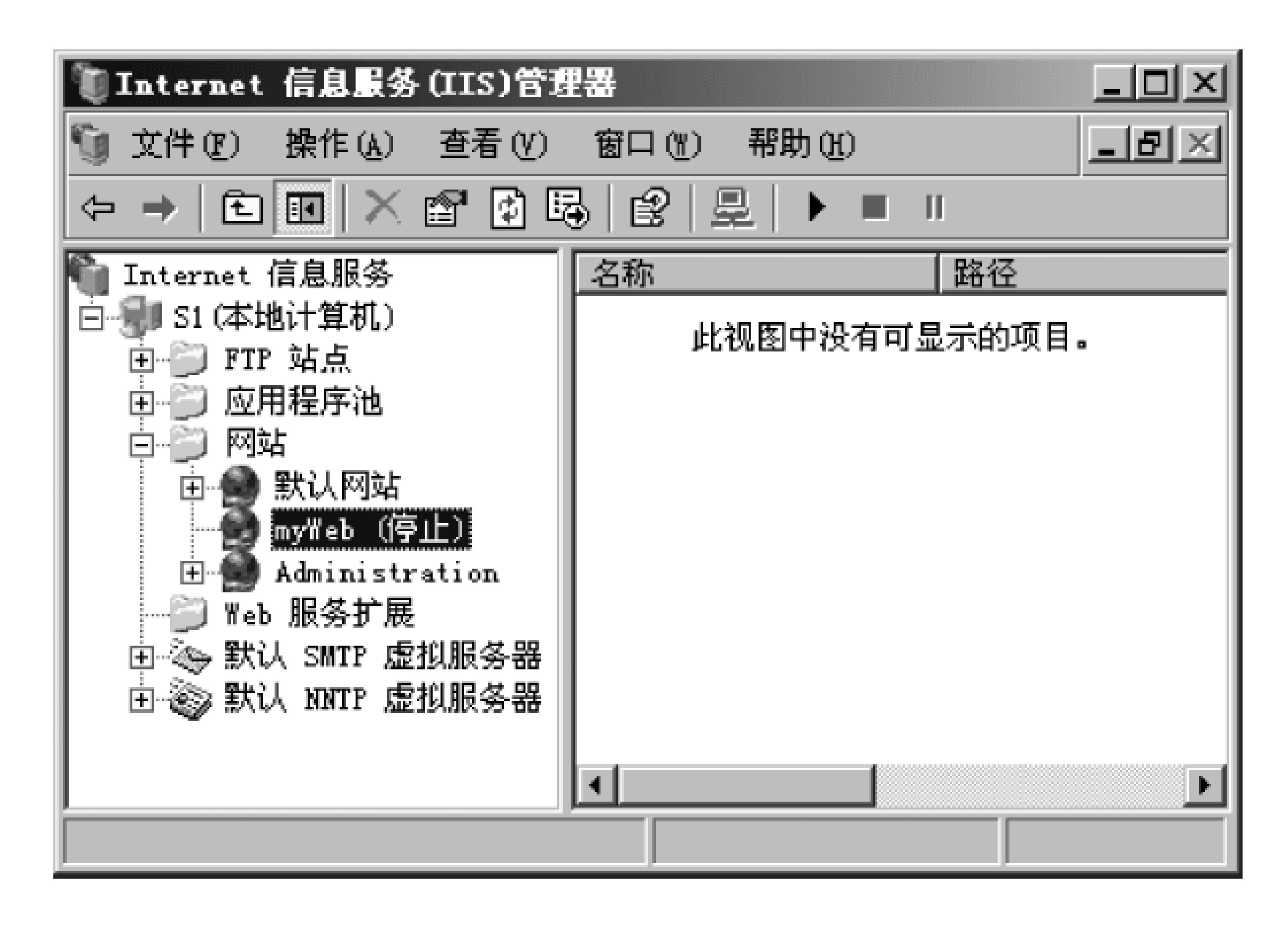


图 2-12 “Internet 信息服务(IIS)管理器”控制台

从控制台中可以看出,新建的站点被标记为“停止”,这是因为它和“默认站点”端口号冲突造成的(均为 80)。要想启动该站点,可以修改站点的端口号,或者修改其他站点属性,比如主机头、IP 地址。如果新站点主目录中没有任何内容,则右侧的窗格显示为空。

2.3.2 Web 站点的启动、停止和暂停

从上面可看到由于新建站点和默认 Web 站点 IP 地址和端口号完全一样,使得新建站点被停止。如果要将停止的 Web 站点启动,右击被停止的 Web 站点,在快捷菜单中,选择“启动”命令,该站点将被启动。如果要停止一个 Web 站点,右击该站点,在快捷菜单中,选择“停止”命令,该站点将被停止。

当管理人员需要维护系统或网页数据时,可以暂停 Web 站点,站点暂停后,它将不接受客户浏览器的连接,等用户工作结束后,再启动该站点。

如果用户试图连接一个暂停的站点,客户端浏览器显示“找不到该页”消息(HTTP 404-未找到文件)。如果试图连接一个停止的站点,客户端浏览器显示“该页无法显示”的消息(找不到服务器或 DNS 错误)。

2.3.3 规划 Web 应用

一个 Web 站点建立后,就意味着一个 Web 应用的开始。所谓 Web 应用,是指在 Internet 环境中,应用程序新的开发和使用模式,它是 B/S 结构下,应用程序的实现形式。一个 Web 网站可以简单地看做一个 Web 应用,它由主目录、子目录及其包含的网页文件、图片文件及其他各类文件,以及相关的数据库构成。

1. 规划网站的文件结构

一个网站，即一个 Web 应用，应该根据用户需求来设计网站的功能，或栏目。为了管理方便，应该根据网站功能对网站文件夹结构进行认真规划。一般情况下，在主目录下往往需要创建多个子文件夹，每个文件夹对应网站的一个功能，存储相关的网页文件。对于一些公用的程序或图片，可以定义单独的文件夹。此外，还可以规划数据库文件夹，存储网站用到的数据库文件，便于整个网站的备份。

对于新建的网站，假设该网站设计有四个主要功能栏目：即时消息、在线聊天、BBS 和个人博客，在主目录下可以分别创建四个文件夹，分别存储开发即时消息、在线聊天、BBS 和个人博客所用到的网页文件，文件结构可以如图 2-13 所示。



图 2-13　站点主目录下的内容组织

2. 文件夹和文件的命名

在开发实践中，为了管理的方便，在命名文件和文件夹时，需要遵循下面几条一般性的命名原则。

(1) 使用名称前缀。因为文件和文件夹的列表通常按照字母排序，因此，可以将功能相近的文件夹或文件，使用相同的名称前缀，从而保证列表时能够挨在一起。

(2) 使用名称后缀或序号。许多功能可以分成几个步骤，每一个步骤可能是一个网页文件，为了管理方便，在命名这些网页文件时，可以在名称后面部分添加序号或后缀，这样可以保证在列表中这些文件是顺序相连的。

例如，注册一个用户，可能分成两个步骤，对应的网页文件名可以为 newuser(注册信息输入页面)、newusersave(数据库操作)。也可以命名为 newuser1、newuser2 等。

(3) 大小写问题。有的 Web 服务器(例如 Tomcat)区分文件夹和文件名大小写，命名时要注意。

(4) 避免中文命名。因为有些 Web 服务器对中文命名支持不好，在命名文件夹和文件名时，尽量避免中文名。

3. 网站首页

传统的应用程序都有一个主用户界面,包含菜单栏、工具栏等,用户通过菜单命令或工具按钮执行特定的程序功能。在B/S结构中,一个Web应用则是从网站首页开始的,相当于传统的应用程序主用户界面。

首页(Home Page)是当客户连接到一个站点时首先看到的Web页面。在设计Web站点的首页时,不仅要考虑页面的外观、栏目布局,更重要的是在页面内容上必须包含可以到各种功能页面的超链接。首页的默认文件名一般为index.htm、default.htm等,首页文件通常需要保存在Web站点的主目录下。

2.3.4 访问Web站点

Web站点是由主目录及其包含的一系列文件夹和网页文件构成的,每个站点都有一个首页文件。用户访问网站即是通过Web浏览器并从站点中下载网页文件的过程,要连接到一个Web站点,应该在浏览器地址栏中输入要下载的页面的URL,一般形式为:

```
http://网址[:端口号][/[路径/文件名]][?参数名=参数值&参数名=参数值…]
```

其中:

(1) 网址可以是域名,也可以为IP地址。端口号对应Web服务器上设定的Web站点TCP端口,默认值为80。如果端口号为80,则在URL中可以省略不写。

(2) 路径,是指相对于Web站点主目录的相对路径,如果不指定路径,则代表站点主目录,站点根即紧接在网址(端口)后面的"/",后面是根下面的子目录。

(3) 文件名。访问一个Web站点,即从Web站点中指定的路径中下载文件,并传输到客户端浏览器进行显示的过程。因此,在URL中需要指定要下载文件的路径和文件名。如果未指定文件名,则代表要下载网站首页文件,首页文件在Web站点属性中设置,并存储在Web站点根目录中。

(4) 参数表。在访问一个网页文件时,特别是带有脚本的网页,有时需要将一些参数传给网页中的脚本程序,这些要传递的参数在文件名后面的"?"后面列出,即一系列的参数名/参数值对。

如果是在IIS服务器计算机上,也可以输入http://127.0.0.1或http://localhost来访问本机上的Web站点。其中,localhost为本机(127.0.0.1)的域名。可以用记事本打开hosts文件(文本文件,无扩展名,存储在\WINNT\system32\drivers\etc文件夹中),看到127.0.0.1的域名为localhost,因此hosts又称为本地域名解析。

2.4 Web站点的配置

在Windows Server/IIS中,当Web站点建立后,还需要对Web站点进行管理,管理Web站点是通过"Web站点属性"对话框来完成的。在"Internet信息服务(IIS)管理器"控制台目录树中,右击站点,选择"属性"命令,打开站点属性对话框,通过站点属性对话框,可完成一个站点的配置和管理。

2.4.1　设置 Web 站点端口号

在 Web 站点属性对话框中，“网站”选项卡列出了网站的一般属性，显示了站点当前的属性设置，如图 2-14 所示。

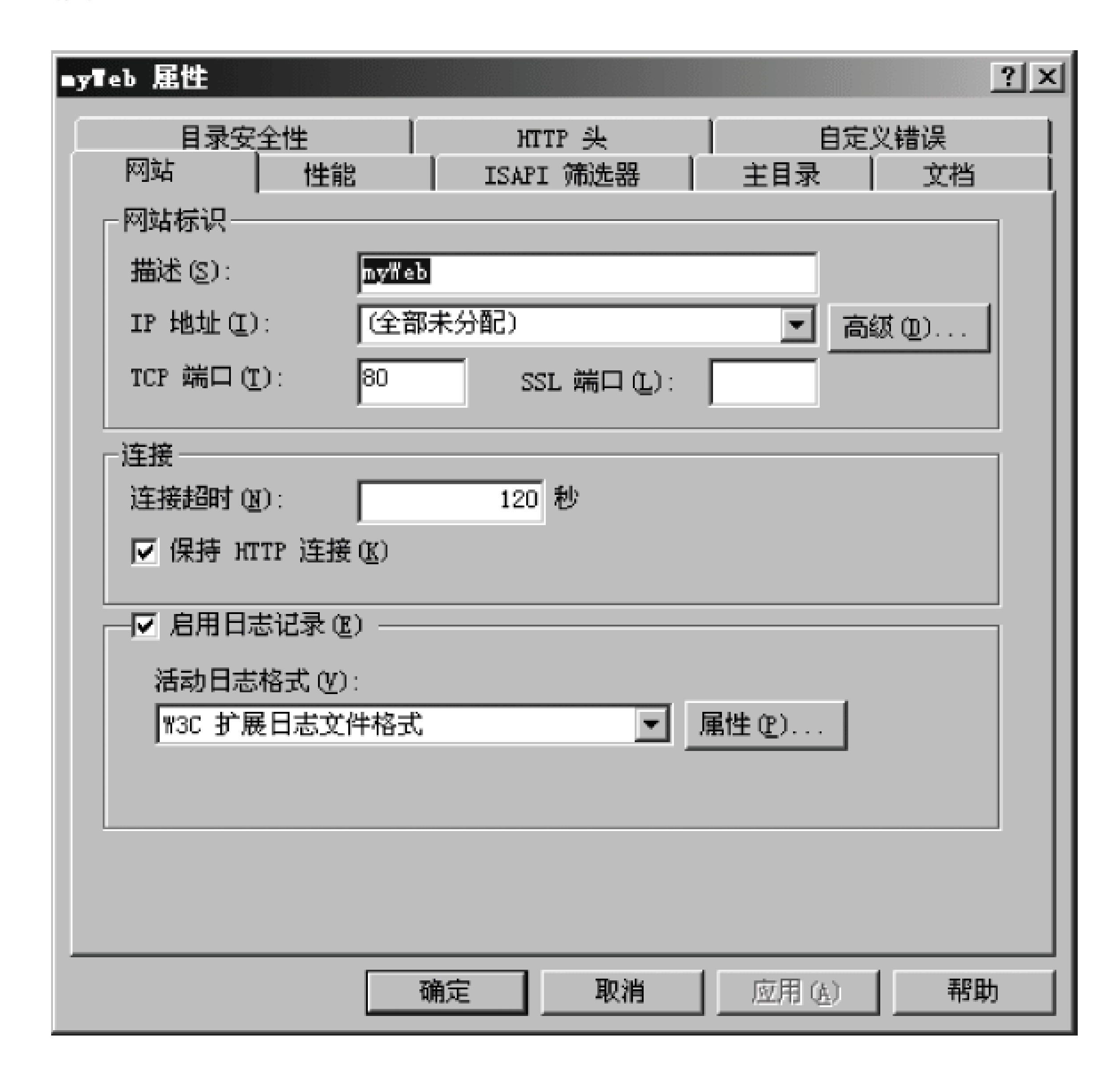


图 2-14　Web 站点属性对话框

在“网站”选项卡中，包括三个选项组的设置。

(1) “网站标识”选项组。该选项组包含以下内容。

① 描述：站点的说明性文字，该文字出现在“Internet 信息服务(IIS)管理器”控制台目录树的节点中，用于标识一个站点。

② IP 地址：设置站点要使用的 IP 地址，如果计算机中设置了多个 IP 地址，可以选择其中一个。如果该服务器上同时运行多个 Web 站点，单击“高级”按钮，可进行进一步的设置。

③ TCP 端口：站点使用的端口号为 80，HTTP 的默认端口为 80。如果设置其他端口，例如将端口号设置为 8001，则客户要访问该网站时，需要在浏览器地址栏中的网址后指定端口号，例如 http://网址:8001/，以指定特定端口的 Web 站点。

(2) “连接”选项组。该选项组包含以下内容。

① 连接超时：如果客户端建立了连接，在连接超过规定的时间没有访问操作，系统将该连接强制断开。

② 保持 HTTP 连接：在 HTML 中，在一个网页中如果包含图片、动画等媒体素材，它们并不是嵌入网页中，而是和网页一样是以独立的文件存在的。如果选中“保持 HTTP 连接”复选框，则当用户下载一个网页时，Web 服务器会将其中的图片等文件一同发送给客户端。

如果取消选中“保持 HTTP 连接”复选框，当网页中包含多个文件连接时，客户端每下

载一个文件就要与 Web 服务器建立一个连接,这将降低 Web 服务器的执行性能。

(3)“启用日志记录”选项组。选中“启用日志记录”复选框将启用 Web 站点的日志记录功能,该功能可记录用户活动的细节并以选择的格式创建日志。启用日志记录后,需要在“活动日志格式”下拉列表框中选择格式。

2.4.2 设置 Web 站点主目录

主目录是一个网站的根,网站的所有文件都保存于主目录及其所包含的子文件夹中,或者通过虚拟目录使用主目录外的物理文件夹。根据客户访问 Web 站点的验证过程,当用户通过身份验证后,接下来,Web 站点会根据站点的权限设置,来决定可以提供给用户的服务,例如从网站浏览网页(下载文件)、上传文件等。

在网站属性对话框中,选择“主目录”选项卡,如图 2-15 所示。

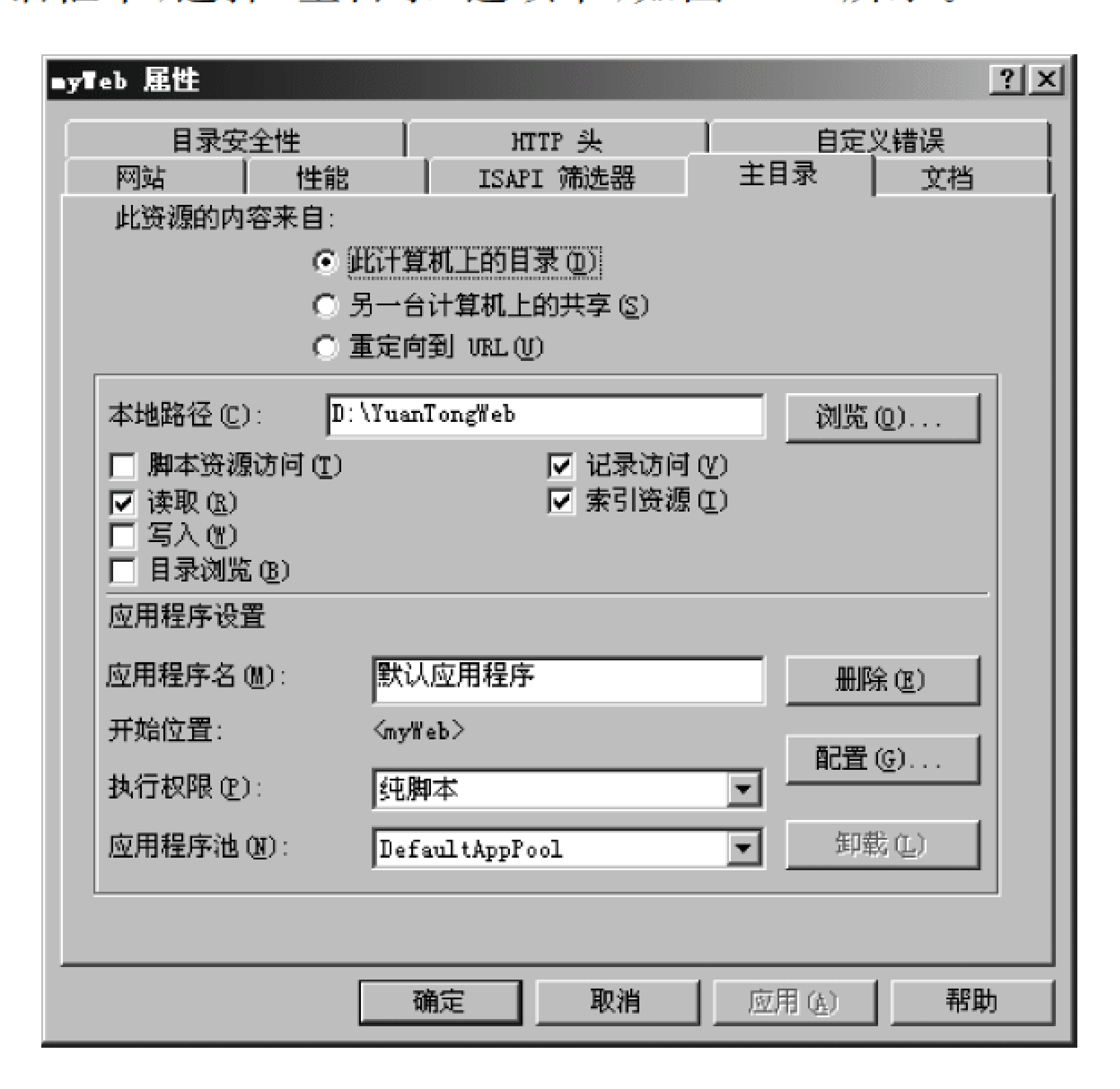


图 2-15 “主目录”选项卡

下面介绍“主目录”选项卡中的部分选项。

(1)访问权限设置。该部分包含以下内容。

① 读取:默认状态下 Web 站点拥有读取权限,即站点提供客户读取服务器上文件的权限,即客户可以从站点中下载文件。

② 写入:允许用户上传文件,或提交表单改变网页内容。

③ 目录浏览:允许用户浏览站点目录,当客户通过浏览器连接到本站点时,如果未指定路径和文件名,默认状态下,Web 服务器将站点的默认文档(首页)发送给客户。此时,如果默认文档不存在,如果选中“目录浏览”复选框,Web 服务器将站点的站点目录列表到客户端浏览器(不显示虚拟目录),如图 2-16 所示。

(2)应用程序设置。应用程序设置可以指定何种应用程序可以在 Web 站点执行。在“执行权限”下拉列表框中,包括“无”、“纯脚本”和“脚本和可执行程序”。如果选择“无”,则

不允许在 Web 站点中运行程序(包括服务器端 ASP 脚本),当浏览一个 ASP 页时,会显示“网页无法显示”,在页面中提示:“您试图从目录中执行 CGI、ISAPI 或其他可执行程序,但该目录不允许执行程序”。选择“纯脚本”,则只能执行 ASP 程序等。选择“脚本和可执行程序”,则所有的应用程序(包括 exe 文件和 dll 库)都可以在 Web 站点上执行。

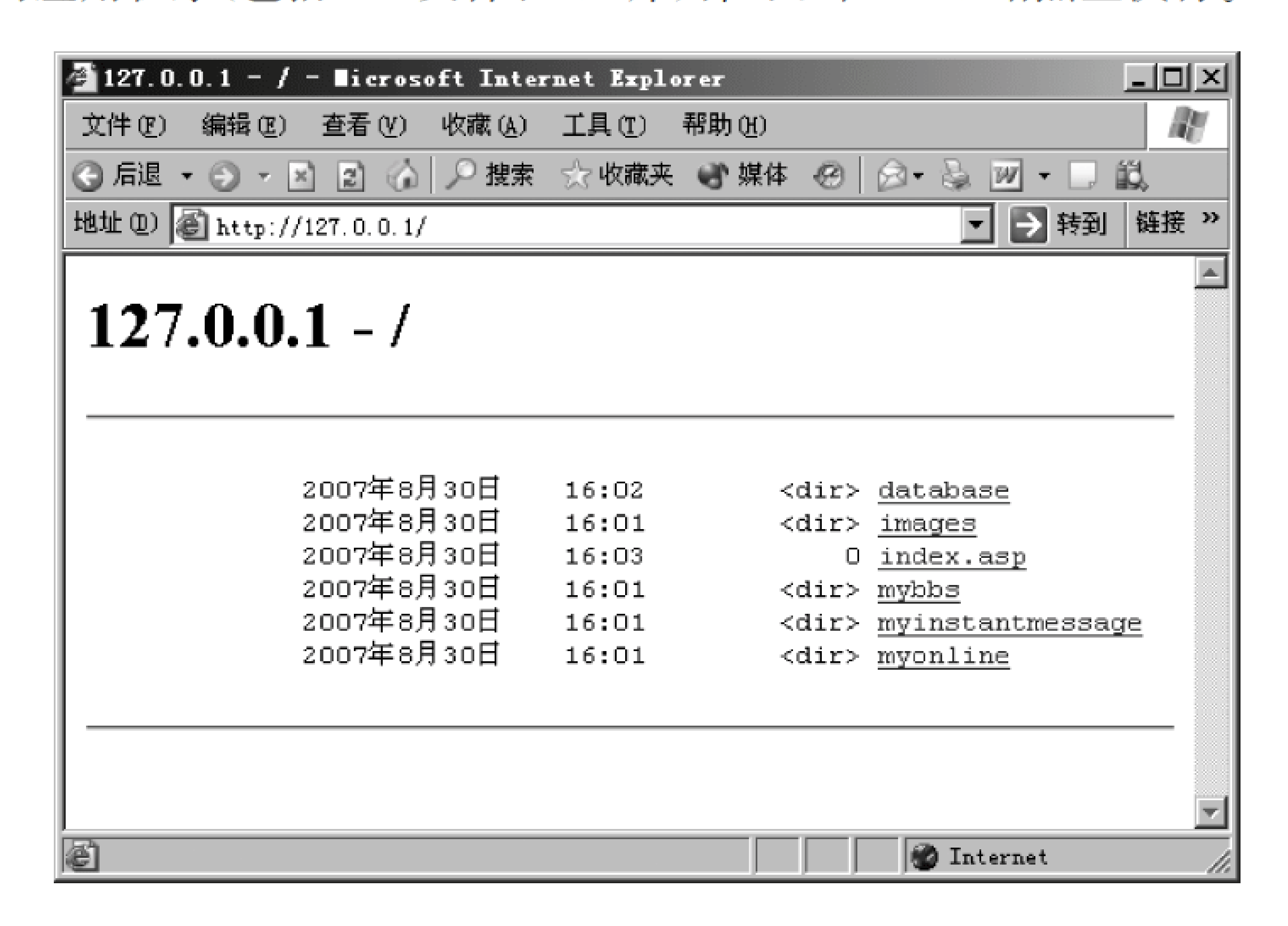


图 2-16　网站目录浏览界面

2.4.3　Web 站点目录安全性配置

Web 站点的安全性设置主要通过“目录安全性”选项卡完成。当用户通过 Web 浏览器发出访问某个页面的请求并且 Web 服务器收到客户的请求后,可以利用 HTTP 请求包获取客户端的信息,同时启动一个验证过程(例如检查用户端的 IP 地址是否受限等)来决定是否将网页传给客户端。

通过验证过程的验证后,Web 服务器会根据网页的类型执行不同的操作。如果网页是 html 类型的,则 Web 服务器将把该网页直接传送到客户端浏览器;如果网页为服务器页(如 asp、jsp 等),Web 服务器将先在服务器端执行该网页文件,然后将执行结果传给客户端浏览器。

要进行 Web 站点的安全性设置,在站点属性对话框中,选择“目录安全性”选项卡,如图 2-17 所示。

下面介绍“目录安全性”选项卡中的相关选项。

(1)“IP 地址和域名限制”选项组。在该选项组中,单击“编辑”按钮,打开“IP 地址和域名限制”对话框,如图 2-18 所示。

在“IP 地址和域名限制”对话框中,选择“授权访问”单选按钮,单击“添加”按钮,可以指定不能访问该站点的 IP 地址。类似地,选择“拒绝访问”单选按钮,单击“添加”按钮,可以指定在拒绝访问中,能够访问该站点的 IP 地址清单。

当用户来自拒绝访问的 IP 地址时,客户浏览器端会收到“您没有权限查看网页”的提示信息。一般情况下,如果网站是公开的,选择“授权访问”单选按钮,单击“添加”按钮,把不被

欢迎的 IP 地址列出。相反,如果网站是一个特殊的站点,只允许部分人访问,则选择“拒绝访问”单选按钮,单击“添加”按钮,把可以访问的 IP 列出。



图 2-17 “目录安全性”选项卡



图 2-18 “IP 地址和域名限制”对话框

(2)“身份验证和访问控制”选项组。当 Web 站点验证了客户端的 IP 地址后,接下来查看该站点是否允许匿名访问。如果站点不允许匿名访问,或者客户端要访问的文件有特殊的 NTFS 限制,此时客户端需要输入用户账户和密码。

当 Web 站点允许匿名访问时,客户端不需要输入账户和密码就可以访问网站的数据,此时 Web 站点会尝试用 Internet Guest Account 账号“IUSER_计算机名称”这个内部账户让计算机登录。要设置匿名访问,在“身份验证和访问控制”选项组中,单击“编辑”按钮,打开“身份验证方法”对话框,如图 2-19 所示。

选中“启用匿名访问”复选框。单击“浏览”按钮,打开“匿名用户账号”对话框可以指定用于匿名访问的匿名用户账号。匿名访问使得每个人都可以使用上述账号访问 Web 网站。如果匿名账户没有足够的 NTFS 权限,系统会根据在“验证访问”选项组中选择的验证方

式，要求用户输入账号和密码，如果未选择任何验证方法，则系统不提示用户输入账户和密码，而是直接拒绝用户对该页的访问。

一般情况下，如果 Web 站点连接到 Internet，选中“启用匿名访问”复选框，即允许匿名访问。如果 Web 站点设置了匿名访问，当客户访问该站点时，仍然出现“输入网络密码”对话框，这是由于匿名账户不拥有要访问的 NTFS 权限造成的。

为了避免遗漏对 Web 主站点的 NTFS 权限设置，导致客户端不能正常地访问所需要的 Web 页，在“Internet 信息服务(IIS)管理器”控制台中，提供了“权限向导”命令。右击站点，在快捷菜单中，选择“所有任务”→“权限向导”命令，启动“IIS 权限向导”。按照向导提示操作，一般取默认值，最后单击“完成”按钮。再来访问该站点，看能否成功。

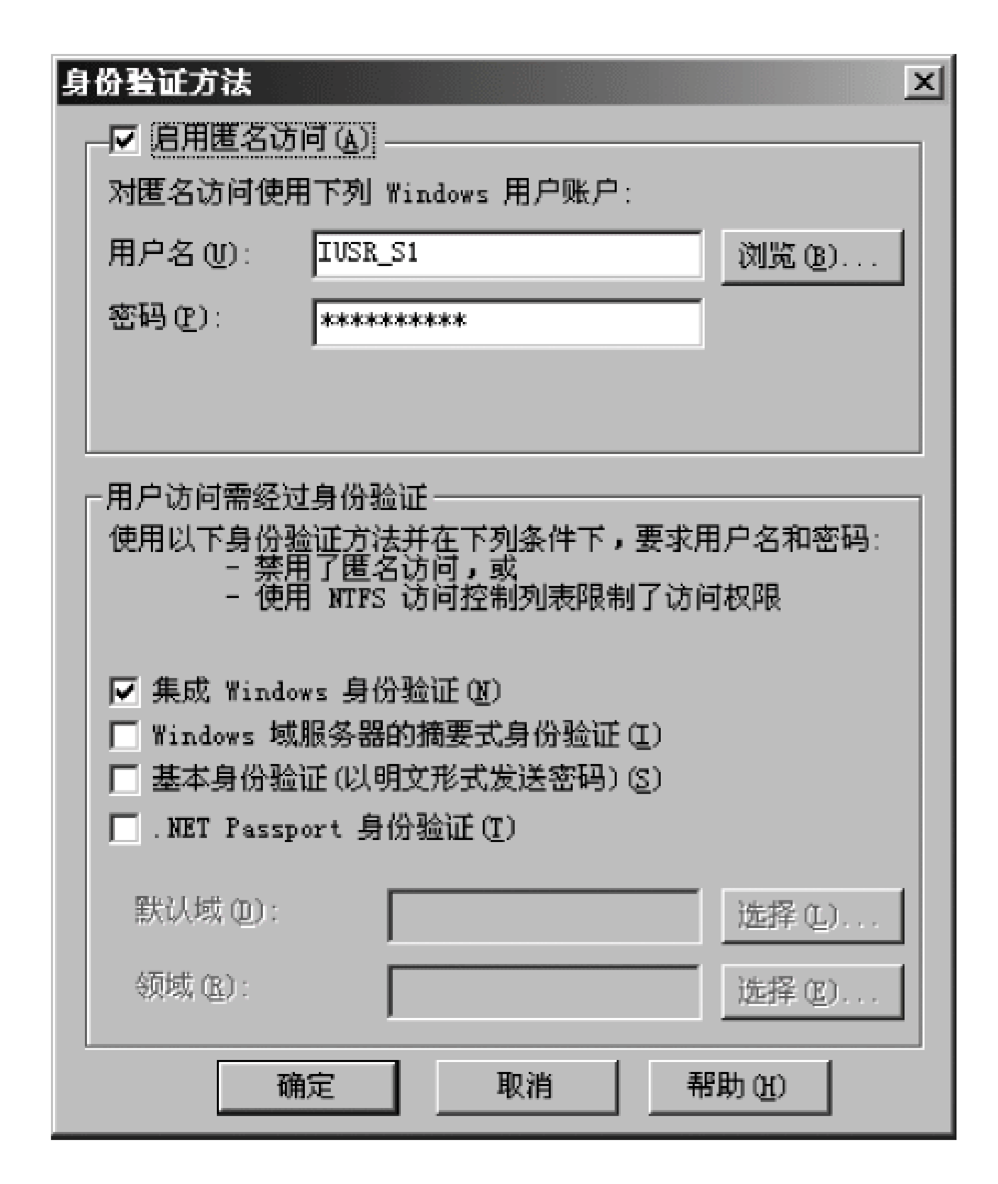


图 2-19　“身份验证方法”对话框

2.4.4　设置 Web 站点默认文档

下面介绍如何设置站点的默认文档，即相当于站点的首页。默认文档可以是 htm 文件，也可以是 asp、jsp 等包含服务端脚本的文件。当用户通过浏览器连接到 Web 站点时，如果没有指定要浏览的文档，Web 站点将默认文档传送给用户浏览器。

在 Web 站点属性对话框中，选择“文档”选项卡，如图 2-20 所示。

选中“启用默认内容文档”复选框，也可以单击“添加”按钮，增加一个新的默认文档，如 index. htm、startpage. htm 等。如果有多个默认文档，系统将把排在前面的文档优先传送给客户浏览器。通过列表下面的“上移”、“下移”按钮，可以改变相应顺序。

如果选中“启用文档页脚”复选框，则服务器在传送要求的网页之前，会在文档的底部插入页角文字，然后再传送。

文档页脚对应一个 htm 文件，这个 htm 文件不应该是一个包含＜html＞＜/html＞、＜body＞＜/body＞等标记的完整的 htm 文件，只能包含文字的大小和颜色设置，例如：

```
<h3 align = right > E - learning 站点</h3>
```

文件可以用 Windows 操作系统中的“记事本”程序编辑,并保存为.htm 类型的文件。



图 2-20 “文档”选项卡

2.4.5 设置 Web 站点 HTTP 头

第 1 章已经介绍了 HTTP 及 HTTP 通信的过程,HTTP 头(HTTP Header)是对现有 http 标准的扩充,有许多复杂的应用,实现客户端和服务端的信息交换。在“HTTP 头”选项卡中,可以对站点做四种设置,如图 2-21 所示。



图 2-21 “HTTP 头”选项卡

选中“启用内容过期”复选框，可以设置此站点页面的内容有效期。Web 服务器将页面发送到客户端时，同时也发送了网页的有效期。

选择“立即过期”单选按钮，则网页内容一下载到浏览器端该页面就过期了。因此，浏览器每次连接到该网站时，无论客户端的本地网页缓存是否存在对应的页面，页面都会被重新下载。它适合于一些显示即时行情，如股市行情的网站。

选择“此时间段后过期”单选按钮后，设置网页的有效期，当浏览器连接到该站点浏览网页时，网页被保存在客户端的缓存文件夹中，时间到后，该网页将自动地从客户端缓存中删除，适合于一些固定时间更新的新闻站点和页面。

“过期时间”单选按钮和“此时间段后过期”类似，用于设置网页的有效期。

在网页浏览中，缓存技术和有效期设置可以提高整个互联网的性能。当用户浏览某个网页时，它首先要检查本地的缓存是否保存了要浏览的页面，如果已经保存该页面，还要检查页面是否过期，如果没有过有效期，则浏览器将直接从本地缓存打开该网页，而不链接 Web 服务器下载页面。如果页面在缓存中不存在，或者存在但已经过了有效期，Web 浏览器将连接 Web 服务器，进行页面下载。对于许多网页(例如一些文件)，其内容并不更新，因此缓存技术可以提高浏览速度，减少互联网流量。

2.5 使用 Apache 和 Tomcat

在 Web 服务器产品中，Apache/Tomcat 是 Apache 软件基金会开发的跨平台的 Web 服务器/应用服务器组合，和微软的 Windows Server/IIS 相比，Apache/Tomcat 具有多种操作系统版本，可以安装在 Windows、Linux、UNIX 等不同的操作系统平台中。此外，由于 Web 服务器还决定了 Web 程序的类型，IIS 及内置的 ASP 引擎支持 asp 和.NET 开发，如果需要开发基于 Java 技术的网站，也需要选择 Apache/Tomcat 服务器。基于上述原因，在商业应用领域，Apache/Tomcat 比 Windows Server/IIS 的应用更加广泛。

2.5.1 Apache 与 Tomcat

Apache 是 Apache 软件基金会开发的 HTTP Server，简称 Apache 服务器，有多个操作系统平台版本，是使用最广的 Web 服务器之一，它可以运行在几乎所有广泛使用的计算机系统平台上，以高效、稳定、安全、免费而著称。Apache 服务器具有以下特性。

(1) 支持 HTTP/1.1 通信协议。

(2) 支持基于 IP 和基于域名的虚拟主机。

(3) 支持多种方式的 HTTP 认证。

(4) 提供用户会话过程的跟踪。

(5) 支持通用网关接口(CGI)。

(6) 集成 Perl 处理模块。

(7) 集成代理服务器模块。

(8) 支持实时监视服务器状态和定制服务器日志。

(9) 支持服务器端包含指令(SSI)。

(10) 支持安全 Socket 层(SSL)。

(11) 支持 FastCGI。

(12) 通过第三方模块可以支持 Java Servlet。

(13) 拥有简单而强有力的基于文件的配置过程。

Tomcat 是 Apache 软件基金会 Jakarta 项目中的一个核心项目,由 Apache、Sun 和其他一些公司及个人共同开发而成,是针对 Apache 服务器开发的 JSP 应用服务器,是 Java Servlet 和 Java Server Pages(JSP)技术的标准实现,由于有 Sun 的参与和支持,最新的 Servlet 和 JSP 规范总是能在 Tomcat 中得到体现,因为 Tomcat 技术先进、性能稳定,是目前流行的 Web 应用服务器。Tomcat 是基于 Apache 许可证下开发的自由软件。可以从 Apache 的官方网站 http://jakarta.apache.org/tomcat/index.html 下载不同的 Apache Tomcat 版本。

可以这样认为,当在一台机器上配置好 Apache 服务器,可利用它响应对 HTML 页面的访问请求。当用户要浏览的页面是 JSP 服务器页面时,Apache 调用 Tomcat,由 Tomcat 执行 JSP 中的服务端脚本程序,将执行结果返给 Apache,然后 Apache 再将结果页面发送到客户端浏览器。实际上 Tomcat 部分是 Apache 服务器的扩展,但它是独立运行的,所以当运行 Tomcat 时,它实际上作为一个与 Apache 独立的进程单独运行。

2.5.2 Apache 的安装与基本配置

Apache 服务器为开源软件,可以从 Apache 官方网站(http://www.apache.org/)下载。在 Apache 官方网站首页中,有一个 Apache Projects 列表,显示 Apache 项目超链接列表,单击 HTTP Server 超链接,将打开 Http Server 项目页面(http://httpd.apache.org/)。

1. 下载 Apache 服务器

在 Http Server 项目页面(http://httpd.apache.org/),选择要下载的 Apache 版本。需要说明的是,版本不一定是最新的,但一定要选择一个稳定的版本,目前使用较广的版本是 Apache HTTP Server 2.2.8。然后,根据操作系统不同选择不同的 Apache 服务器。

对于 Windows 平台,有两个可选的版本,分别是 Win32 Binary without crypto (no mod_ssl)和 Win32 Binary including OpenSSL 0.9.8g。OpenSSL 为开放安全套接层(Secure Socket Layer,SSL)协议,可以在 Internet 上提供秘密性传输,包含密码算法库、SSL 协议库以及应用程序,目前的版本为 0.9.8g。选择 Win32 Binary including OpenSSL 0.9.8g 下载,将服务器文件 apache_2.2.8-win32-x86-openssl-0.9.8g.msi 下载到本地计算机(4.8MB)。

目前,Apache for win32 使用 msi 的形式发布,.msi 文件类型是一种可以安装的程序包文件,双击带.msi 扩展名的文件时,操作系统将.msi 文件与 Windows 安装程序关联并运行客户端安装程序服务 Msiexec.exe,从而使 Windows 环境下安装 Apache 变得非常简单。

2. Apache 的安装

当 Apache 服务器下载后,可以按照下列步骤完成 Apache Web 服务器的安装和配置。

(1) 双击 Apache 的安装文件 apache_2.2.8-win32-x86-openssl-0.9.8g.msi,执行安装向导,如图 2-22 所示。

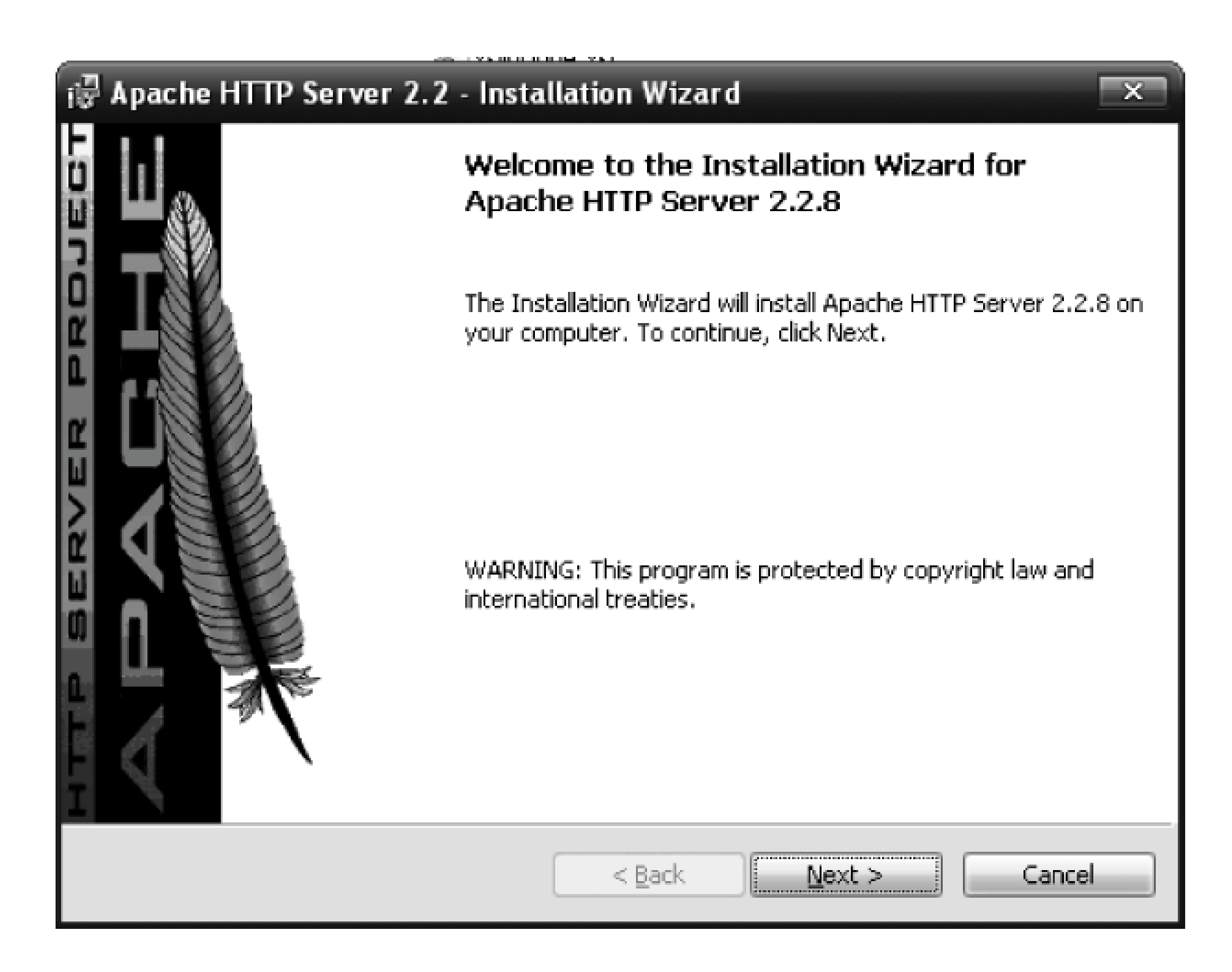


图 2-22　Apache 服务器的安装

（2）单击 Next 按钮，打开 Server Information 界面，如图 2-23 所示。

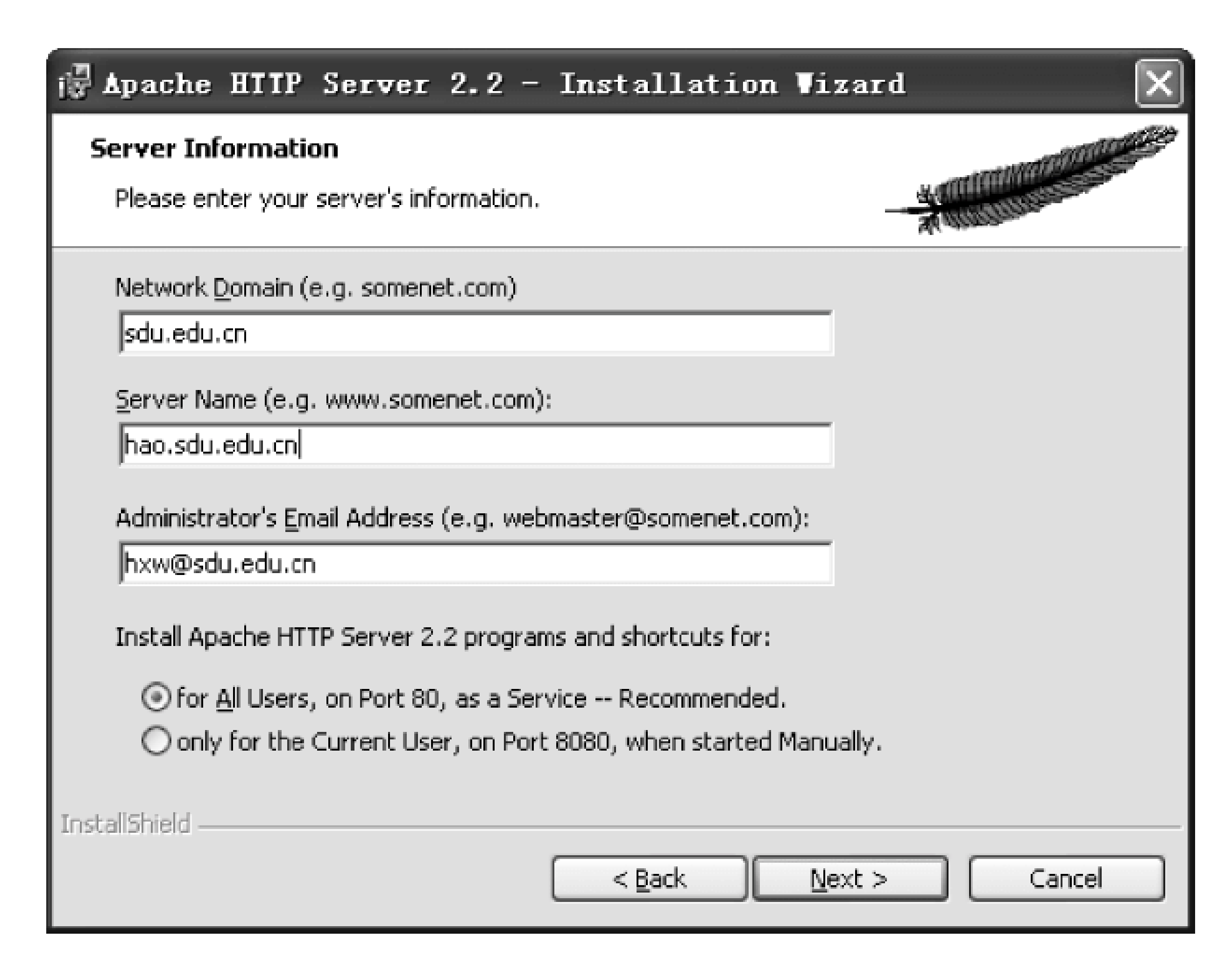


图 2-23　Apache 安装向导输入信息界面

（3）在 Apache 的安装过程中，需要输入网站域名（即网站所在的 DNS 域的名字）、服务器的域名（即服务器主机名＋域名）、网站管理员的 E-mail，按照网站的实际情况填写。如果是个人用户，可能没有上述数据，可按格式填一下临时的名字。然后，单击 Next 按钮，选择安装类型（Typical 或 Custom）。

（4）单击 Typical 按钮，按照向导提示操作，选择安装路径，直至安装完成。安装结束后，指定安装目录的文件夹结构如图 2-24 所示。

（5）Apache 服务安装成功后，在 Windows“开始”菜单中增加 Apache HTTP Server 2.2 程序组。同时，在“控制面板”、“管理工具”文件夹下，双击“服务”图标，显示 Apache 已经启动，以后 Apache 将作为一项服务，随着机器的启动而自动运行。

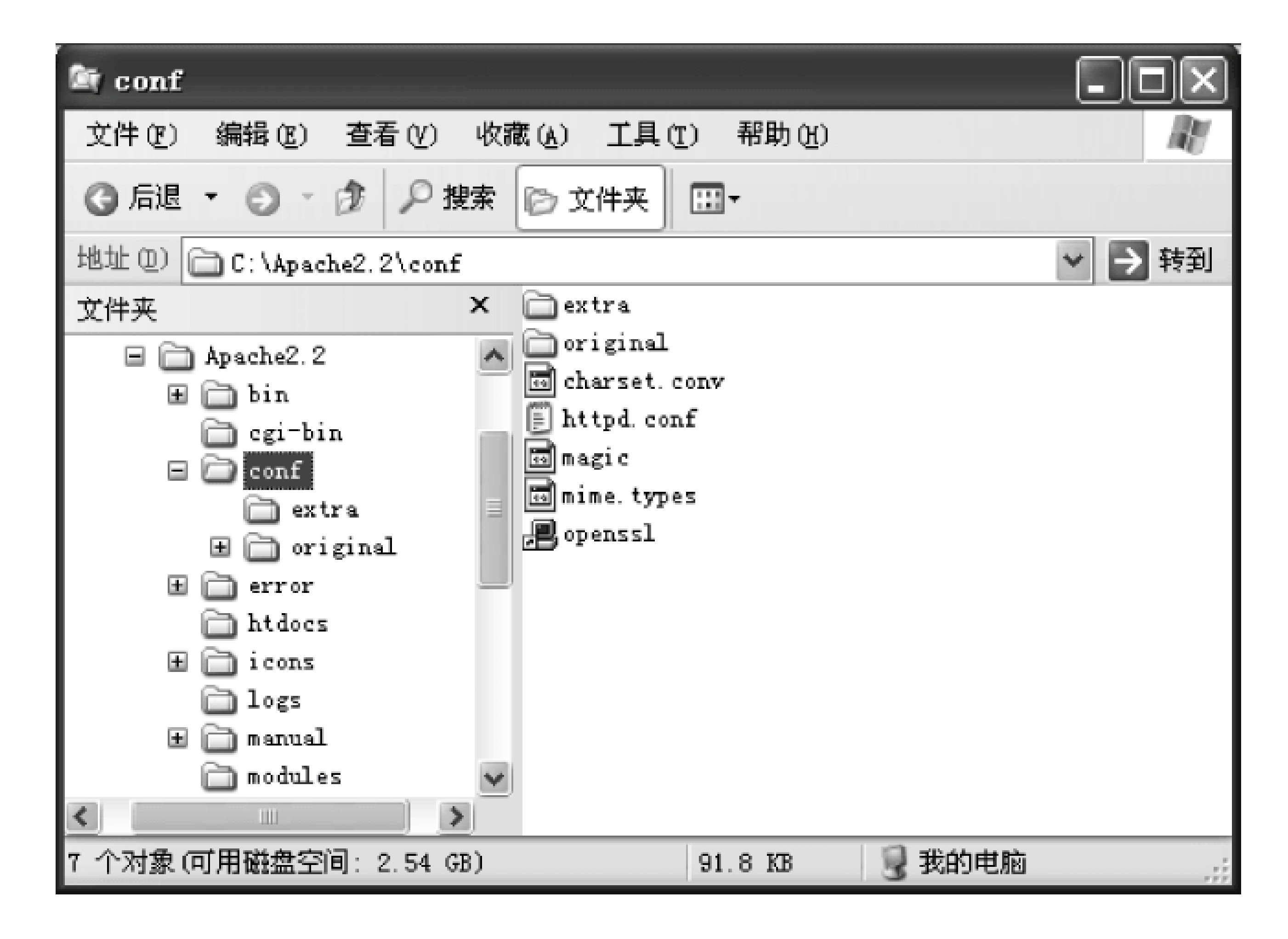


图 2-24　Apache 安装目录文件夹结构

(6) 不需要重新开机,Apache 会自动启动,在 Windows 任务栏的右侧显示 Running all Apache Services 图标。此时运行浏览器程序,在地址栏中输入 http://localhost/或 http://127.0.0.1/,看到默认的 Apache 首页,显示 It works。

需要说明的是,如果计算机上已经安装了 IIS,输入 http://localhost/后将首先显示 IIS 的默认站点。此时,可以按照 2.3.2 节的介绍停止 IIS 中的 Web 服务器,或者给 IIS 中的 Web 服务指定一个不同的端口号(不同于默认的 80 端口)。然后再输入 http://localhost/即可显示 Apache 服务器设置的页面。此时还可以通过 http://localhost:端口号/继续打开 IIS 中的默认 Web 站点。

3. Apache 的配置

Apache 的主配置文件为纯文本格式的 httpd.conf,它的存储位置为\Apache\Conf\。随着 Apache 版本的发展,趋向于使用单一的配置文件 httpd.conf 来存放所有的配置指令,如客户访问信息、记录认证信息和虚拟服务器信息等。

Apache 配置选项采用的是指令模式,配置指令设定各种参数的值,例如 DocumentRoot 设置服务器 Web 页面的根目录。也可以灵活地设置多个基于 IP 或基于域名的虚拟 Web 服务器,这些 Web 虚拟服务器可以各自定义独立的 DocumentRoot 配置指令。而 LoadModule 指令则用来指定加载不同的模块来实现对 Apache 服务器功能的扩充。这些新功能大多是提供服务器端对脚本技术的支持,比如 Perl、PHP 等。Apache 结合使用 ApacheJServ 可以实现对 Java Servlet 及 JSP 的支持。

用记事本打开它,可以看到这些配置文件都以文本方式存在,其中“#”为 Apache 的注释符号,可以在“记事本”程序中选择“编辑”→“查找”命令,在弹出的对话框中逐一输入下面要配置的关键字,并进行相应配置。此外,打开 Windows 的“开始”菜单,执行“程序”→Apache HTTP Server 2.2→Configer Apache Server→Edit the Apache httpd.conf Configuration File 命令,将打开记事本,显示 httpd.conf 文本文件,进行 Apache 的配置。

(1) 配置 DocumentRoot 及目录访问权限。指定网站路径,即主页放置的目录。默认

路径一般是 Apache 安装目录下的一个子目录，例如：

```
DocumentRoot "C:/Program Files/Apache Software Foundation/Apache2.2/htdocs"
```

根据需要，设置站点的主目录，例如可以在此处将其设定为"D:/GSL3.0"，打开主页时，默认打开的文档就直接去该目录下查找了。

设置用户主目录后，重启 Apache，在浏览器中输入 http://127.0.0.1/访问 Apache 站点，此时返回一个 403 的错误：

```
Forbidden
You don't have permission to access / on this server
```

上述错误信息是因为目录访问权限设置造成的。此时，打开 Apache 的配置文件 httpd.conf，逐行检查主配置文件 httpd.conf。代码如下：

```
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>
```

修改 Deny from all 为 Allow from all。

(2) 配置 DirectoryIndex。即设置站点默认显示的主页，一般情况下，在此处还可以加入“Index.htm Index.php Index.jsp”等。注意，每种类型之间都要留一空格。

上面两步设置完成后，启动浏览器，输入 IP 即可访问自己的 Web 站点。还可以在该文件的 ServerName 处定义域名，在 ServerAdmin 处输入 E-mail 地址。以上两条就是在安装时选择配置的，以后可以在此处修改它们的属性。

此外，如果要拒绝一部分人访问该 WWW 站点，可以到 Apache 的安装目录下找到 Access 文件，输入要禁止的 IP 地址即可。

2.5.3 Tomcat 服务与 Servlet/JSP 规范

Tomcat 是当前使用最为广泛的 Servlet/JSP 服务器，它是 JavaSoft 和 Apache 开发团队合作计划(Apache Jakarta Project)的产品。通常情况下，Tomcat 作为 Apache 服务器的扩展，它是一个应用服务器，用于运行 JSP 页面中的 Java 程序，是 servlet 和 jsp 的容器。

在第 1 章的 Java 技术中，已经介绍了 Java 技术的构成，即 Java 技术包括三个部分：Java 程序设计语言、JDK 和 Java 虚拟机。因此，要保证 Tomcat 运行 Java 代码，在 Tomcat 运行环境，必须得到 Java 的支持，这就是说 Tomcat 的运行需要 Java 运行环境的支持。

在 Tomcat 中，内置了一个简单的 HTTP Server，因此单独安装 Tomcat 也能够保证一个 Web 站点的运行。从开发的角度没必要用 Apache 和 Tomcat 配合，Tomcat 自己完全可以应付。也就是说，不需要安装 Apache 服务器，单独使用 Tomcat 即可运行 Web 应用。但是，Tomcat 仅仅对 JSP 程序体现出比较好的执行效率和性能，对于静态页面的处理速度远不如 Apache。为了提高 Web 系统的整体性能，需要将 Apache 和 Tomcat 进行整合配置。

可以从 Apache 网站 http://tomcat.apache.org/下载所需要的 Apache Tomcat 版本，

本章选择最新的 apache-tomcat-6.0.14.exe(集成实现了 Servlet2.5 和 JSP2.1 标准),讲解其具体的安装和配置。

2.5.4 安装 Java 运行环境

Tomcat 需要 Java VM(JRE)(Java Runtime Environment,Java 运行环境),即 Java 虚拟机的支持,因此,在安装 Tomcat 以前需要安装 JRE。JRE 可以单独安装,也可以随 Java 开发包 JDK 一起安装。安装 JRE 后,在安装 Tomcat 时会自动监测到。

Java 技术中的 Java 运行环境包括两个主要的部分:Java 开发工具包和 Java 运行环境。它们是基于 Java 技术开发和运行的基础环境。在 Windows 平台上,Java 环境安装完成后需要手工进行相应的环境变量配置,才能正确地工作。

1. 什么是 JDK 和 JRE

在安装 Java 环境以前,需要介绍几个概念。

在 Java 技术中,大家经常看到 JDK、J2SDK 和 JRE 等概念,有时候会产生迷惑,三者是一种什么关系呢?

JDK 是 Sun 公司早期的 Java 软件开发工具包(Java Develop Kit,JDK),包含所有编写、运行 Java 程序所需要的工具:Java 基本组件、库、Java 编译器、Java 解释器、小应用程序浏览器,以及一些用于开发 Java 应用程序的程序等。从 JDK 1.2 起,Sun 在命名时开始使用 Java2,这就是 J2SDK 了,又分为企业版(Enterprise Edition)J2EE、标准版(Standard Edition)J2SE 以及面向嵌入式和移动计算等领域的 J2ME(Micro Edition)三个不同的版本,详细说明参见 1.4.1 节。

JRE,顾名思义是 Java 程序运行所需要的环境。所谓跨平台就是各种平台都有一个中间代理,这就是 JRE。一般采用 Java 技术开发的软件都需要安装 JRE,所以 Sun 公司就单独提供了 JRE 安装文件,以供 Java 应用程序发布时所用。

以上 Java 软件都可以从 Sun 公司的 Java 网站①(http://java.sun.com)上获取,网站上分别提供了 J2EE SDK、J2SE SDK 以及 Java VM(JRE)各种版本的下载。

2. 安装 JDK 和 JRE

http://java.sun.com/提供了 J2SDK 和 JRE 的集成安装和单独安装文件,用户可以免费下载。目前较新,同时比较稳定的版本是 JDK6。根据开发和应用的不同,可以选择企业版或标准版。下面以 J2SE6 为例,介绍 JDK 和 JRE 的安装过程。

(1) 登录 Sun 官方网站 http://java.sun.com/,在常用现在区域(Popular Downloads),单击 Java SE 超链,显示 Java JDK 和 JRE 下载界面;选择 JDK6 Update 3,下载文件为 jdk-6u3-windows-i586-p.exe,包含 JDK6 和 JRE。

(2) 接下来进行 JDK6 和 JRE 的安装过程。双击 jdk-6u3-windows-i586-p.exe 文件,运行 JDK6 安装向导,显示许可协议,打开自定义安装界面,如图 2-25 所示。

---

① Sun 公司被 Oracle 公司收购后,原有 Sun 公司的域名保留,但会被自动地映像到 Oracle 公司的站点。

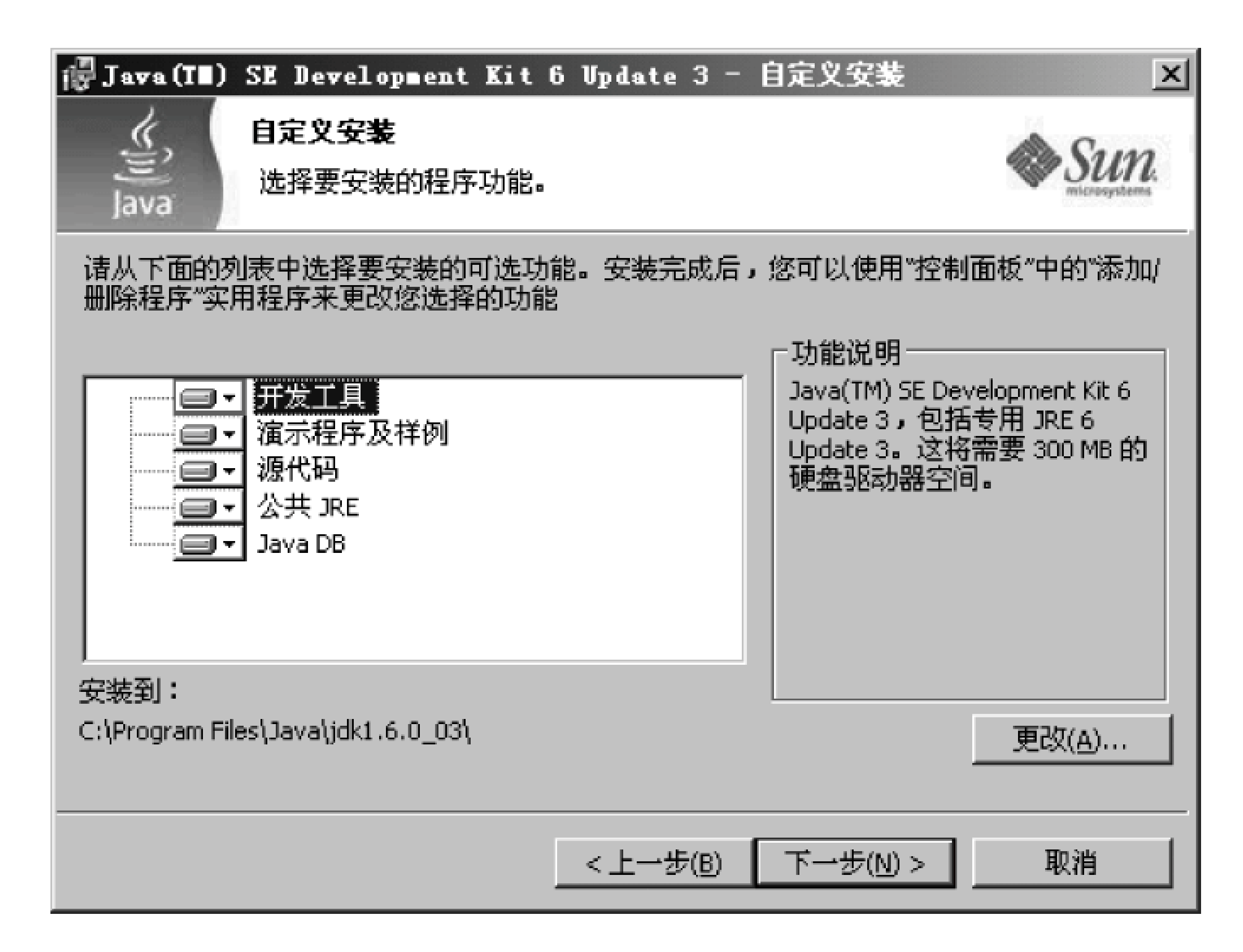


图 2-25　Java2 SDK 标准版安装向导界面

(3) 按照向导提示将 Java 开发环境安装到计算机中，默认的文件夹为 C:\Program Files\java\jdk1.6.0_03。为了下一步环境变量设置的方便，通常需要修改默认安装目录，例如直接安装在 C:\Java 目录下，即 C:\Java\jdk1.6.0_03\，这样便于环境变量的设置。

(4) 由于 jdk1.6.0_03 已经包含 JRE，如果机器尚未安装 JRE，则在安装 jdk1.6 时，JRE 将一并安装。安装过程也需要指定安装路径。和安装 JDK 同样的原因，可以设置 JRE 的安装目录为 C:\Java\jre1.6.0_03\。

(5) 当 JDK 和 JRE 安装完成后，安装程序在 C 盘中建立相应的文件夹结构，存储相应的 Java 运行环境，文件夹结构如图 2-26 所示。



图 2-26　安装 JDK 和 JRE 文件夹结构

(6) 按照向导提示安装完成后,在“控制面板”中显示一个咖啡杯图标。双击该图标,将打开“Java 控制面板”。

可以通过“控制面板”中的“添加/删除程序”删除已经安装的 jdk/jre。

3. Java 环境变量设置

在 Windows 中,JDK 安装完成后,开发和运行 Java 程序,还需要进行相应的环境变量设置,以保证 Java 程序对 JDK 中类库的引用。需要进行的环境变量设置包括设置 JAVA_HOME 和 CLASSPATH 环境变量、更新 PATH 路径设置三个部分。

为了检查 JDK 安装程序是否已经正确地设置了环境变量,可以使用 set <环境变量> 来检查,具体办法是:

在 DOS 提示符下,通过 set <环境变量>命令显示环境变量的配置情况。JDK6 安装完成后,环境变量设置检查结果显示如图 2-27 所示。

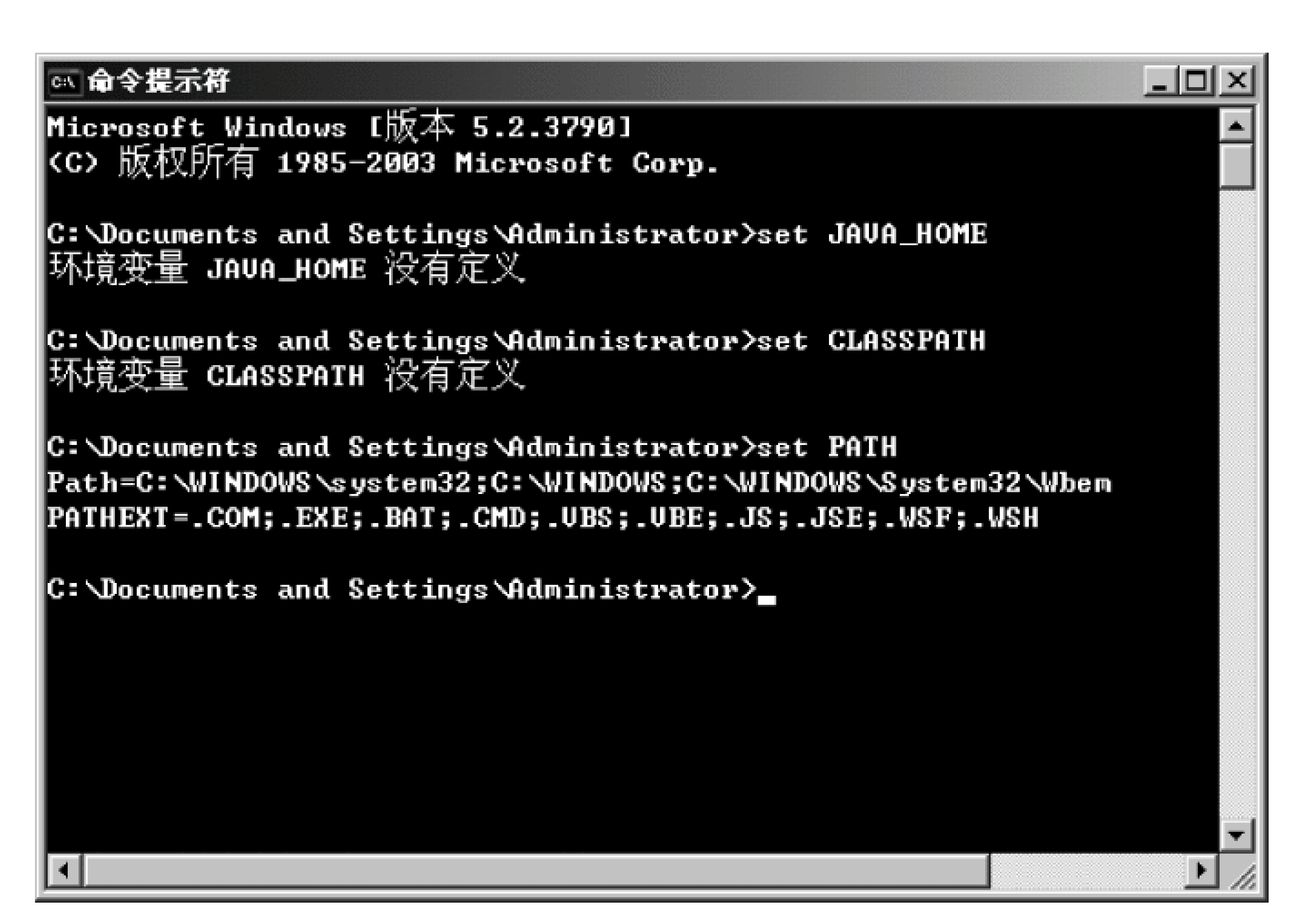


图 2-27 检查系统环境变量设置

如果安装程序没有设置 Java 运行环境需要的环境变量,应该进行手工设置。根据上述的 JDK 安装路径(图 2-26),设置内容如下:

```
set JAVA_HOME = C:\Java\jdk1.6.0_03
set CLASSPATH = .; %JAVA_HOME%\lib(“.”代表当前路径)
PATH = %PATH%; %JAVA_HOME%\bin; %JAVA_HOME%\jre\bin
```

各环境变量功能如下。

(1) JAVA-HOME 表示 Java 的安装目录,在其他环境变量中使用。

(2) CLASSPATH 定义 Javac 搜索类的路径,它记录 Java 编译器和解释器所需要的类所在的路径。即使是用户自己创建的类,也应该添加到 CLASSPATH 中,这样比较麻烦,所以在 CLASSPATH 中添加了一个当前目录(即“.”)。这样,当转到用户所在的目录的时候,由于 Javac 编译生成的用户类保存在当前路径,必须把当前路径加到 CLASSPATH 中,这样 Java 解释器才能够找到用户的类。有时候,会看到 CLASSPATH 中包含一个.jar 等

压缩的 class 文件[1]，把它加入到 CLASSPATH 中，Java 环境可以读取该文件。

(3) PATH 变量是系统搜索可执行程序的路径，其中，Java 编译器(javac. exe)保存在%JAVA_HOME%\bin 中，Java 解释器(java. exe)保存在%JAVA_HOME%\jre\bin 中，要在任何路径下使用 javac. exe 和 java. exe，必须将上述路径定义在操作系统的 Path 环境变量中。

说明：在 CLASSPATH 和 PATH 环境变量的配置中，%...%表示一个操作系统的环境变量。“；”用于分割不同的目录路径，也就是说，如果要设置多个查找路径，路径之间需要用分号(;)分开，“.”代表当前路径。

要设置上述环境变量，需要通过控制面板中的“系统”程序来完成，具体步骤如下：

在 Windows“控制面板”中，双击“系统”图标，打开“系统属性”对话框。选择“高级”选项卡，如图 2-28 所示。单击“环境变量”按钮，打开“环境变量”对话框，如图 2-29 所示。

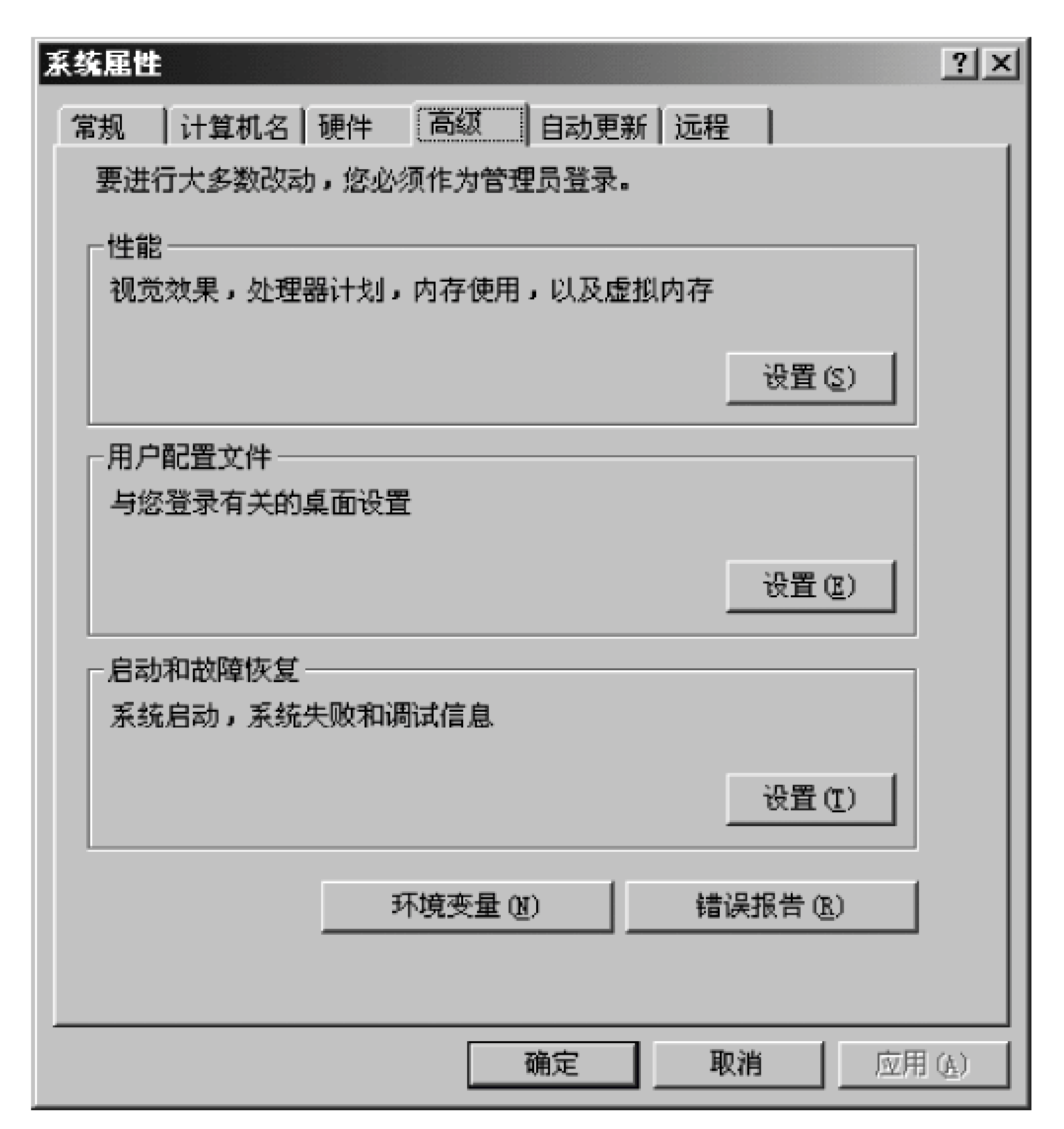


图 2-28 “系统属性”对话框



图 2-29 “环境变量”对话框

在“环境变量”对话框中的“系统变量”选项组中，可以新建环境变量，或者对已经存在的环境变量进行修改。

(1) 设置 JAVA_HOME 环境变量。在“系统变量”选项组中，单击“新建”按钮，打开“新建系统变量”对话框，从中输入要新建的系统变量以及变量值，如图 2-30 所示。

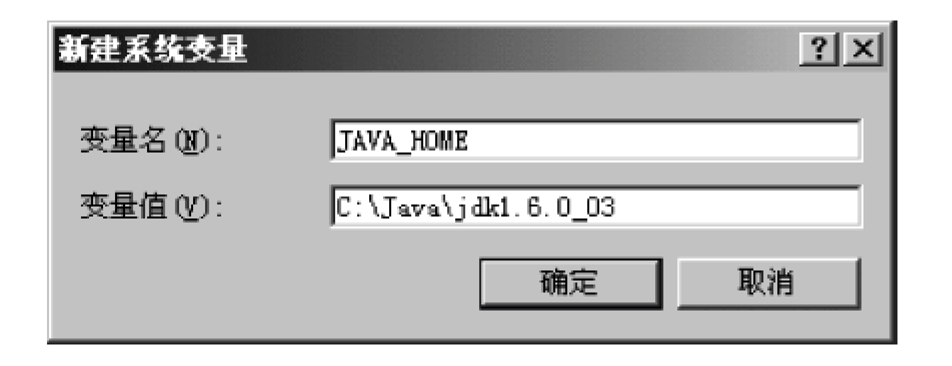


图 2-30 新建系统变量 JAVA_HOME

输入完成后，单击“确定”按钮。

(2) 设置 CLASSPATH 环境变量。用同样的

① jar 的全称是 JavaTM Archive (JAR) file，是 java 存档文件，主要压缩存储 Java 的 class 文件。Jar 是 JavaJDK 中的命令，可以在 DOS 提示符下，使用 jar -help 命令显示 jar 的使用方法。

方法,新建系统变量 CLASSPATH,如图 2-31 所示。

(3) 更新 PATH 路径设置。在"环境变量"对话框中的"系统变量"选项组中,选择 PATH 环境变量,单击"编辑"按钮,在原有 PATH 基础上,增加";%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin",如图 2-32 所示。

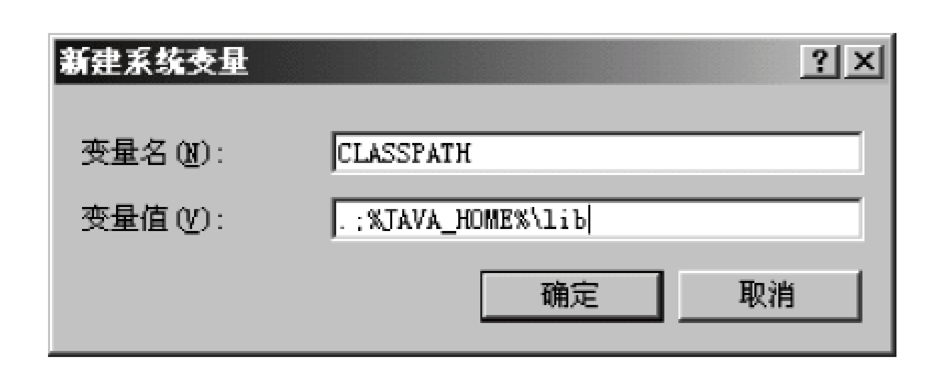


图 2-31 新建系统变量 CLASSPATH



图 2-32 更新系统变量 PATH

4. 测试 Java 运行环境

设置完成后,重新启动计算机,使上述设置生效。然后在 DOS 提示符下,依次输入下述命令来检查环境变量的设置情况:

```
c:\> echo %java_home%
c:\> echo %classpath%
c:\> echo %path%
```

也可以通过 set<环境变量名>命令来检验上述设置。如果设置正确,可以输入下列命令检查 Java 的运行是否正常。

```
c:\> java -version
c:\> javac
```

输入上述命令后,运行结果如图 2-33 所示。

如果能运行 Java 编译命令 javac,表明 Java 的环境变量设置没有问题。接下来可以用一个简单的 java 程序来测试 J2SDK 的安装,代码如下:

```
public class Test
{
  public static void main(String args[]) {
      System.out.println("Hello,My Java program ");
  }
}
```

创建文件夹 D:\MyJava,将上述程序代码保存在该文件夹下,文件名为 Test.java(和类名一致,包括大小写)。然后打开 DOS 命令提示符窗口,转到 Test.java 所在目录 D:\MyJava,然后输入下面的命令:

```
javac Test.java
java Test
```

如果显示"Hello,My Java program",表明 Java 环境安装成功,用 dir 命令可看到生成一个 Test.class 的文件;否则需要仔细检查环境配置情况。

```
命令提示符
C:\Documents and Settings\Administrator>javac
用法: javac <选项> <源文件>
其中, 可能的选项包括:
  -g                         生成所有调试信息
  -g:none                    不生成任何调试信息
  -g:{lines,vars,source}     只生成某些调试信息
  -nowarn                    不生成任何警告
  -verbose                   输出有关编译器正在执行的操作的消息
  -deprecation               输出使用已过时的 API 的源位置
  -classpath <路径>            指定查找用户类文件和注释处理程序的位置
  -cp <路径>                   指定查找用户类文件和注释处理程序的位置
  -sourcepath <路径>           指定查找输入源文件的位置
  -bootclasspath <路径>        覆盖引导类文件的位置
  -extdirs <目录>              覆盖安装的扩展目录的位置
  -endorseddirs <目录>         覆盖签名的标准路径的位置
  -proc:{none,only}          控制是否执行注释处理和/或编译。
  -processor <class1>[,<class2>,<class3>...]要运行的注释处理程序的名称; 绕过默认
的搜索进程
  -processorpath <路径>        指定查找注释处理程序的位置
  -d <目录>                    指定存放生成的类文件的位置
  -s <目录>                    指定存放生成的源文件的位置
  -implicit:{none,class}     指定是否为隐式引用文件生成类文件
  -encoding <编码>             指定源文件使用的字符编码
  -source <版本>               提供与指定版本的源兼容性
  -target <版本>               生成特定 VM 版本的类文件
  -version                   版本信息
  -help                      输出标准选项的提要
  -Akey[=value]              传递给注释处理程序的选项
  -X                         输出非标准选项的提要
  -J<标志>                     直接将 <标志> 传递给运行时系统


C:\Documents and Settings\Administrator>java -version
java version "1.6.0_03"
Java(TM) SE Runtime Environment (build 1.6.0_03-b05)
Java HotSpot(TM) Client VM (build 1.6.0_03-b05, mixed mode, sharing)

C:\Documents and Settings\Administrator>_
```

图 2-33　检验 Java 运行情况

2.5.5　Tomcat 的安装和配置

首先登录 Tomcat 官方网站 http://tomcat.apache.org/，在 Download 区域，单击 Tomcat 6.x 超链接，显示 Tomcat 6.x 的下载界面，在 Tomcat 6.0.14 的二进制代码发布 (Binary Distributions)区域，单击“Windows Service Installer (pgp，md5)”超链接，即可下载 Tomcat 安装程序，文件名为 apache-tomcat-6.0.14.exe。

1. 安装步骤

执行 Tomcat 安装程序 apache-tomcat-6.0.14.exe，启动安装向导，按照向导提示执行下面的步骤。

(1) 选择要安装的 Tomcat 组件，如图 2-34 所示。

在安装类型下拉列表框中，选择完全安装(Full)，Tomcat 将作为 Windows 服务器的服务直接启动。

(2) 选择安装的物理路径，默认路径为 C:\Program Files\Apache Software Foundation\Tomcat 6.0，如图 2-35 所示。

为下一步配置环境变量方便，我们修改安装路径为 C:\Tomcat 6.0。

(3) 进行 Tomcat 的基本配置，包括 HTTP 端口，Tomcat 的默认值为 8080，可以修改

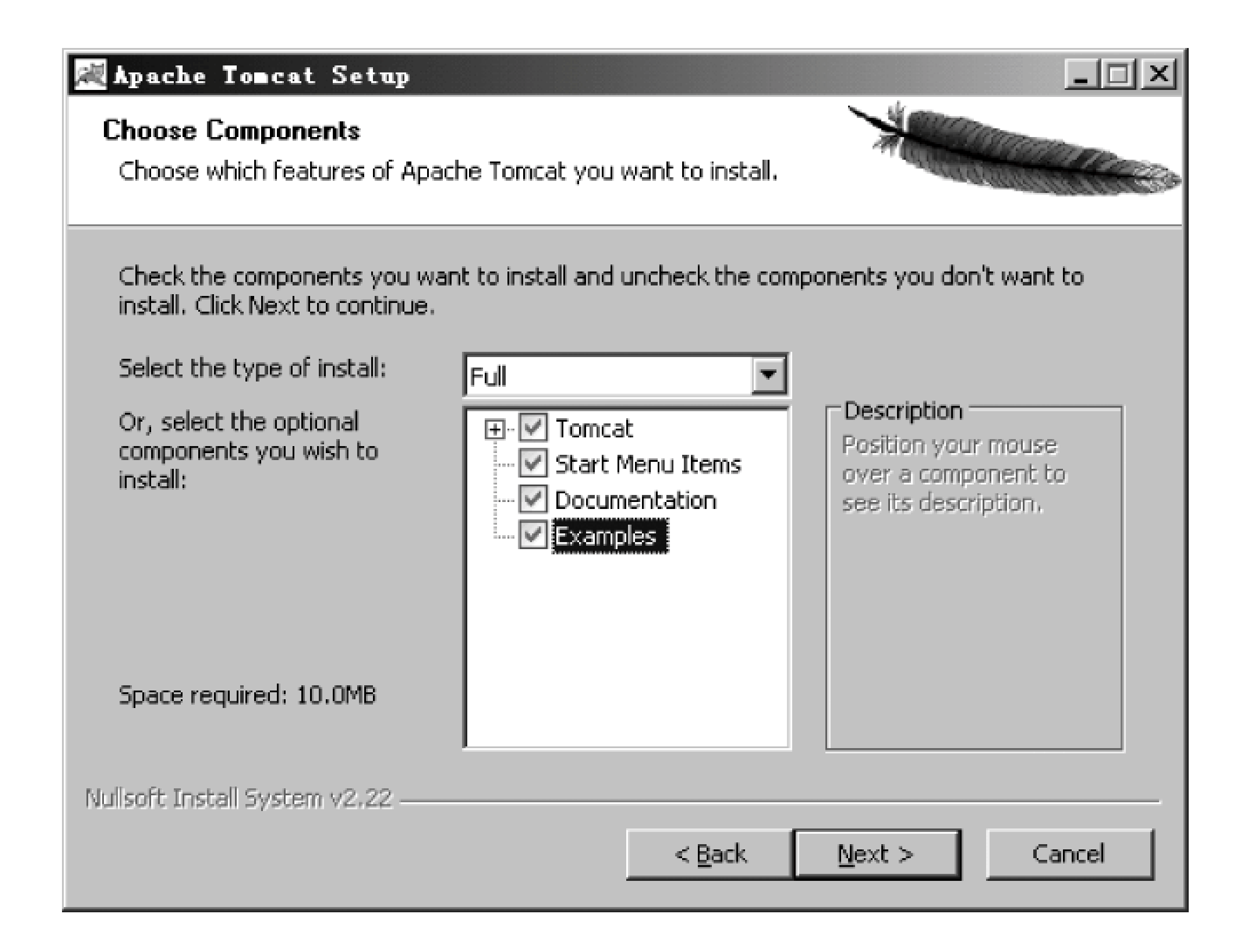


图 2-34　Tomcat 安装向导界面

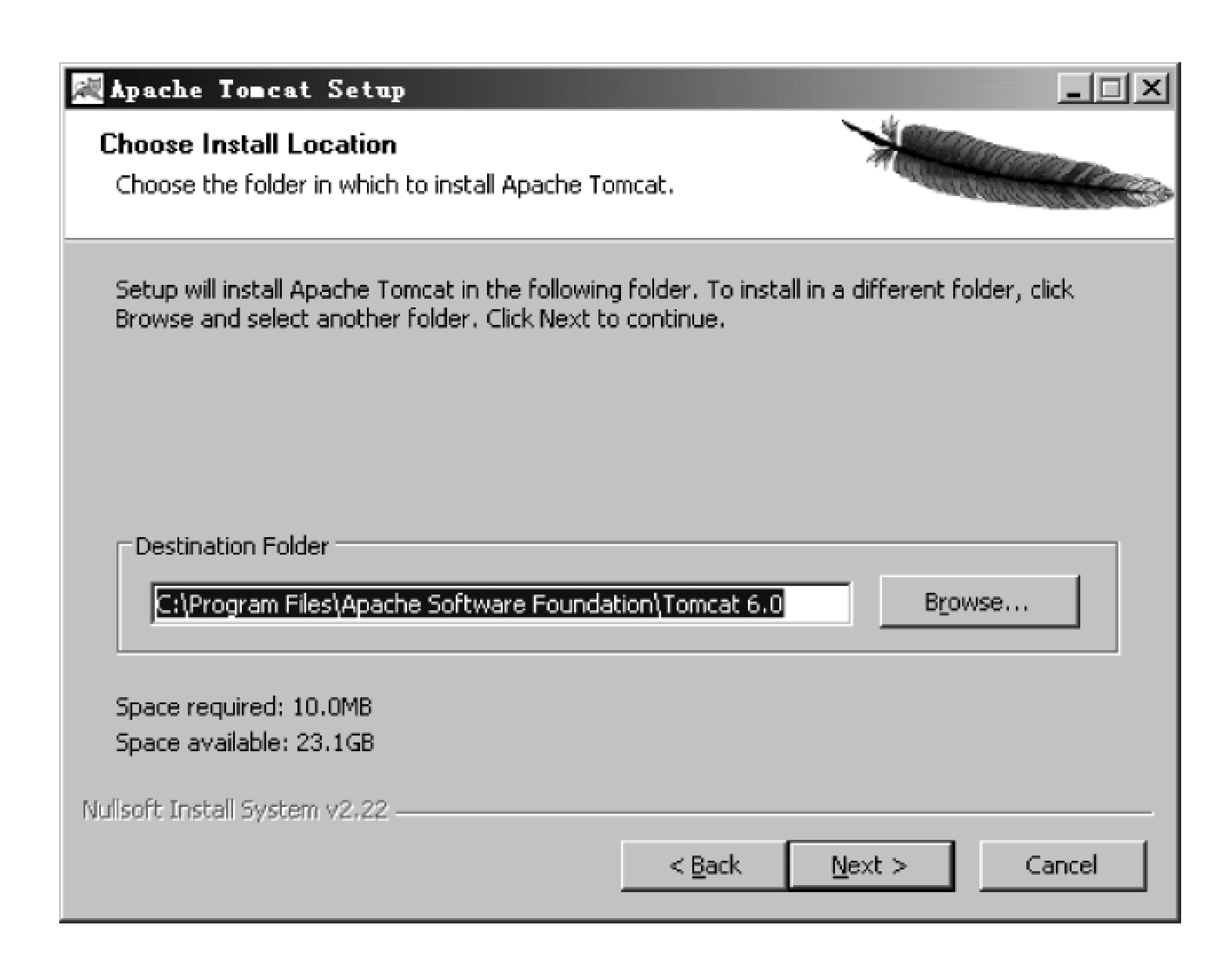


图 2-35　选择 Tomcat 安装路径

为 80；管理员的登录名和密码，默认登录名为 admin，密码可以为空，如图 2-36 所示。

对于 Tomcat 的端口号，因为默认的 Web 服务器配置是 Apache＋Tomcat，因此，Tomcat 的默认端口设置并不是 80，而是 8080，Apache 也正是通过 8080 这个端口和 Tomcat 进行通信的。因此，如果服务器配置了 Apache，此时，不要修改 Tomcat 端口号，使用默认的 8080。如果服务器不安装 Apache，只是用 Tomcat 本身，可以修改端口号为 80 或其他。

(4) 选择安装 Java Virtual Machine 的物理路径。如果已经成功配置完毕 JDK（含 JRE），向导直接指向 J2SDK 中安装的 JRE 目录，例如 C:\Java\jre1.6.0_03，如图 2-37 所示。

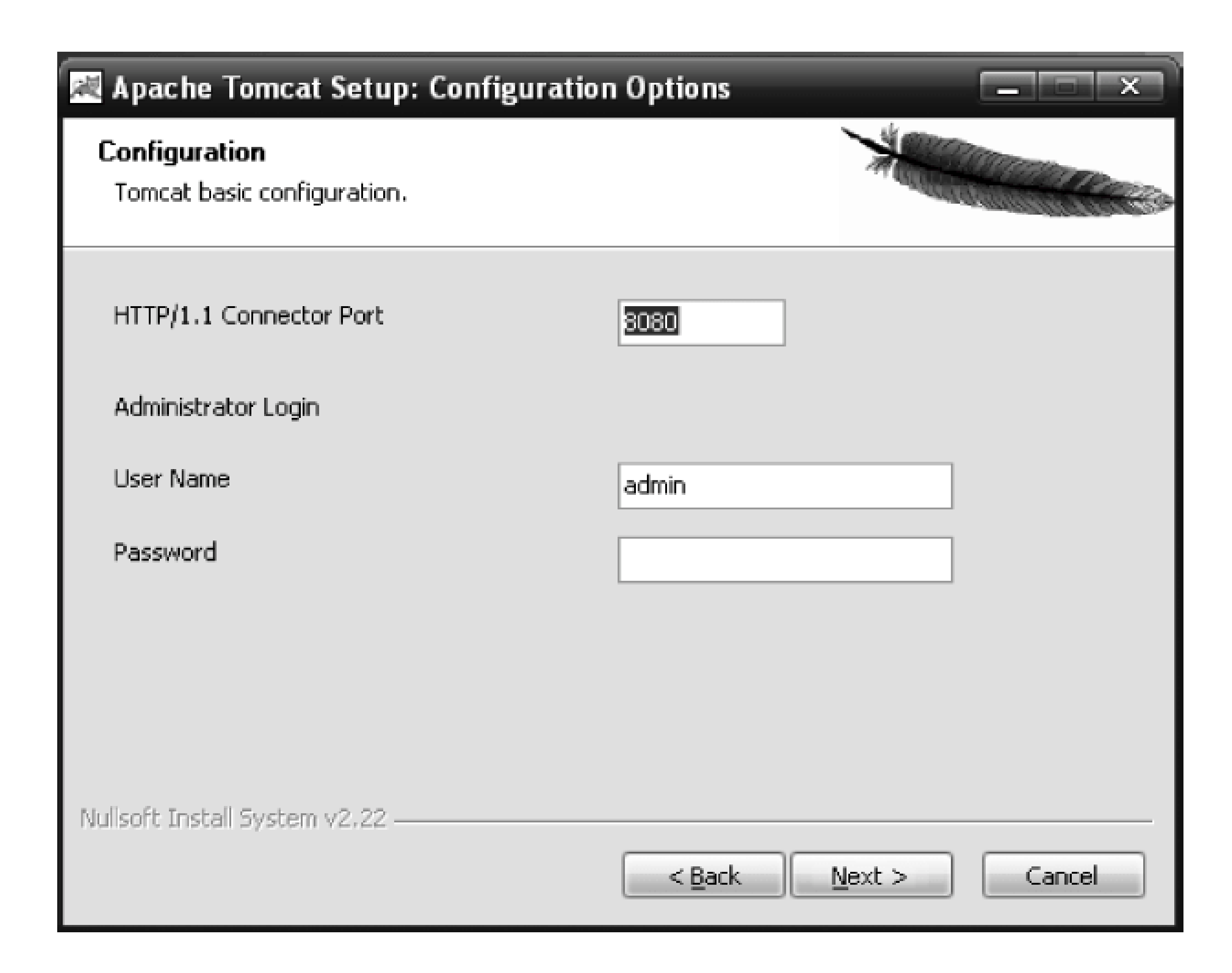


图 2-36　设置 Tomcat 服务端口号

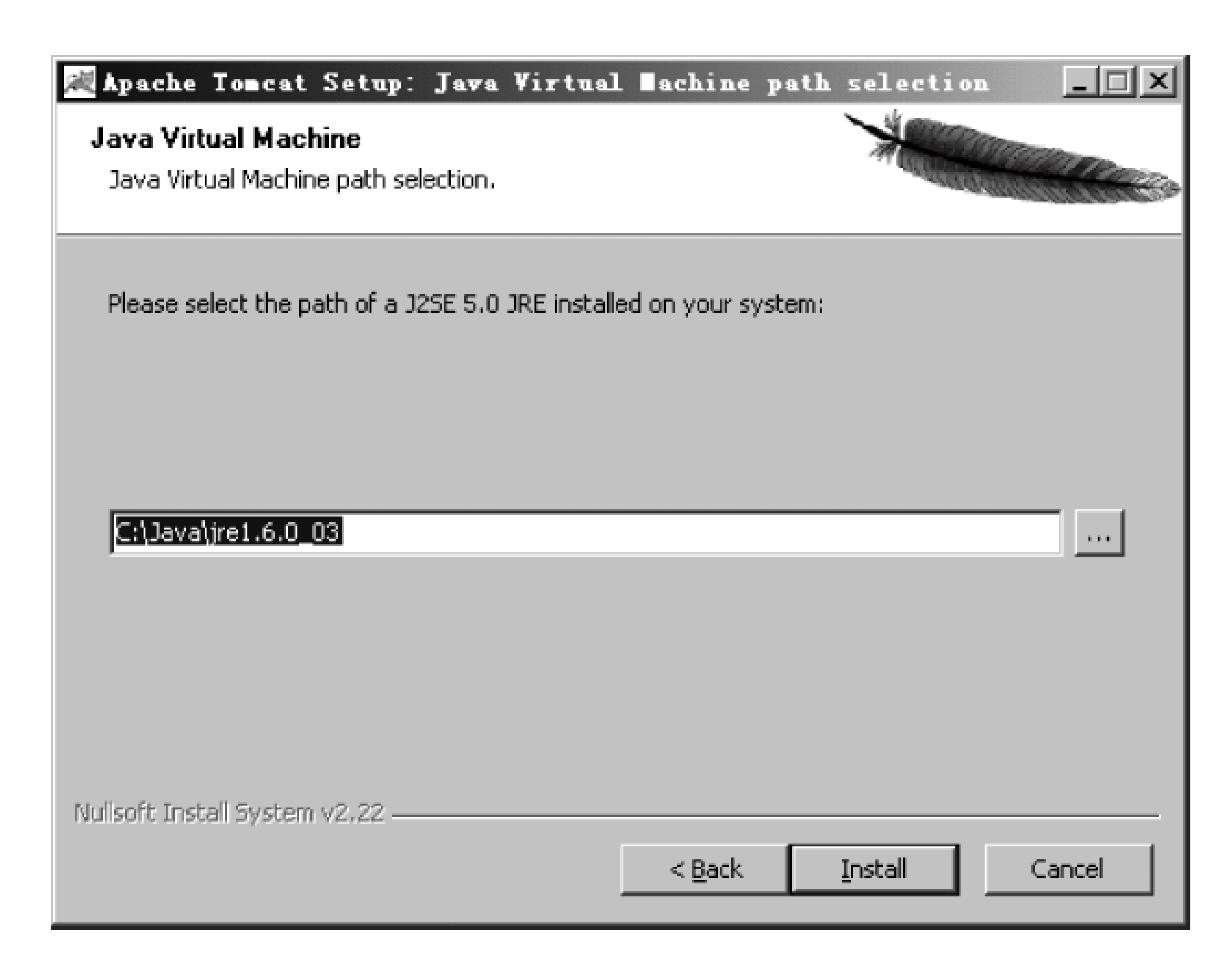


图 2-37　指向 JRE 路径

单击 Install 按钮，开始安装，向导将把有关的文件复制到相关的目录下，并自动启动 Tomcat。Tomcat 安装完成后，在“开始”菜单的“程序”组中，将增加 Apache Tomcat 6 程序组。

(5) 测试安装是否成功。打开 IE 浏览器，在地址栏中输入 http://127.0.0.1:8080/（或 http://localhost:8080/），如果出现如图 2-38 所示的界面，则表明 Tomcat 安装成功。

Tomcat 安装完成后，安装程序将建立相应的目录，所建立的目录结构如图 2-39 所示。

不同的 Tomcat 版本，安装完成后的文件夹结构不同。Tomcat 6.0 的文件夹结构比 Tomcat 5.5 简单，各文件夹及其功能说明如下。

(1) bin 目录，主要存放 Windows 平台上启动和关闭 Tomcat 的脚本。

(2) lib 目录，存放 Tomcat 服务器以及所有 Web 应用都可以访问的 jar 文件。为了在

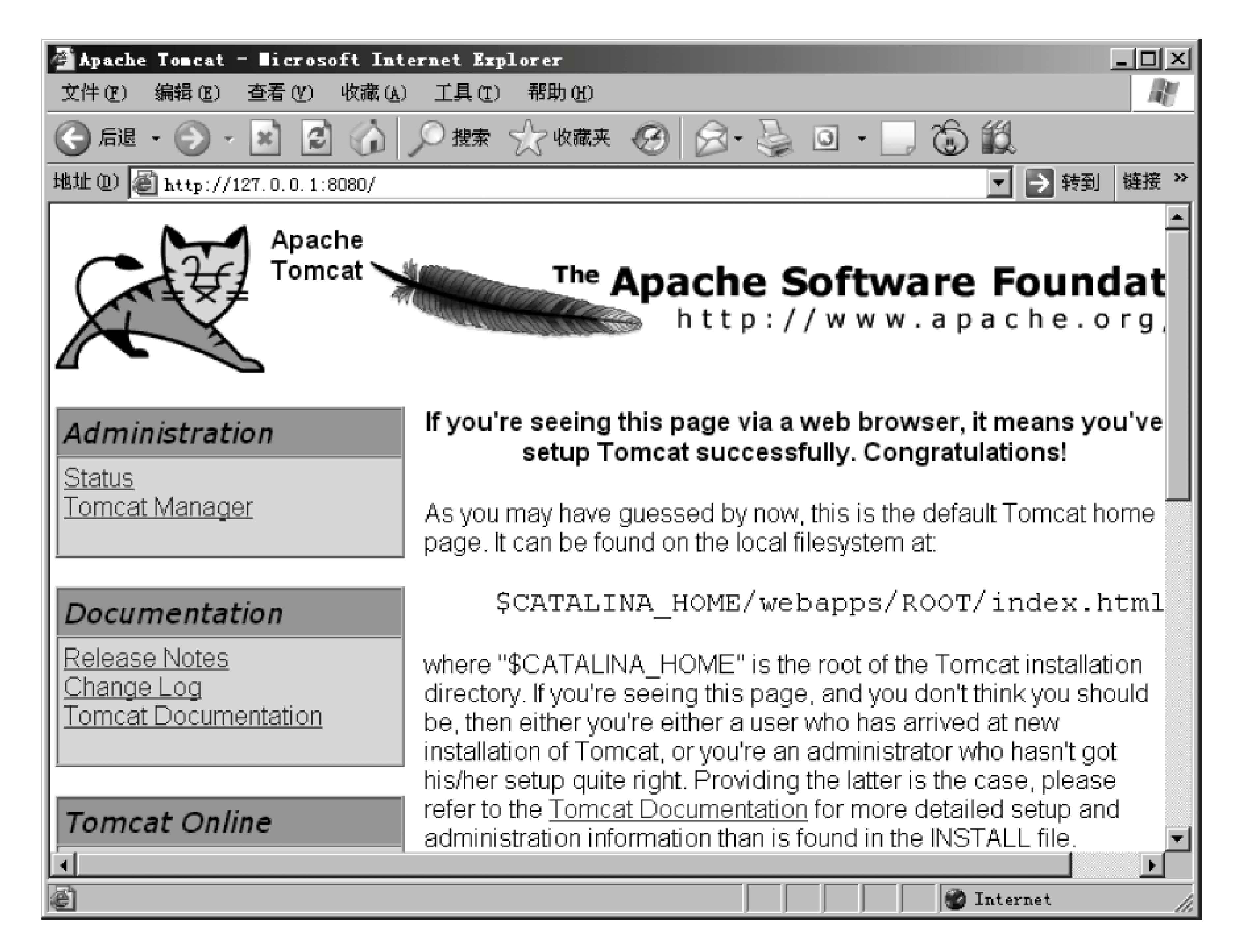


图 2-38 Tomcat 安装成功界面

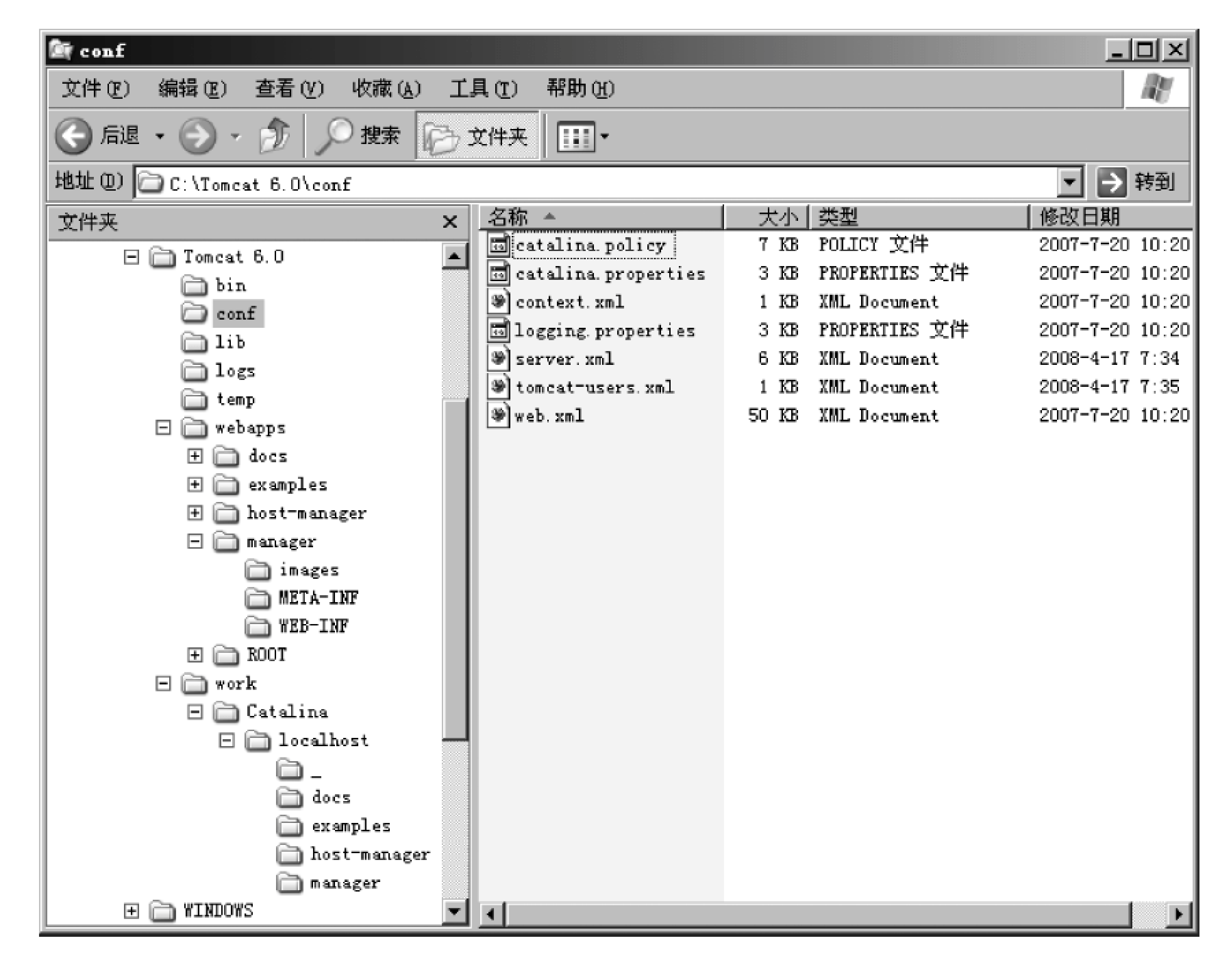


图 2-39 Tomcat 安装目录结构

java 环境下能够正确编译 servlet 文件，最好把 lib 目录中的 jsp-api. jar 和 servlet-api. jar 复制到 J2SDK 的安装目录的 lib 子目录（即 c:\ Java\jre1. 6. 0_03\lib）内，同时，需要在 CLASSPATH 环境变量中也增加这两个. jar 文件（即在原变量后面输入“;c:\java\ jre1. 6. 0_03\lib\jsp-api. jar; c:\ java\jre1. 3. 0_03\lib\servlet-api. jar”）。

（3） conf 目录，存放 Tomcat 服务的配置信息文件，其中最重要的是 server. xml 和 web. xml。server. xml 是 Tomcat 的主配置文件，可以在其中配置 Web 服务的端口、会话过期时间、虚拟目录、虚拟主机等。web. xml 为不同的 Tomcat 配置的 Web 应用设置默认值。另外，在其/Catalina/localhost 子目录下还可以设置网站虚拟目录和根路径信息等。

（4） logs 目录，存放 Tomcat 执行时的 Log（日志）文件。

（5） temp 目录，存放 Tomcat 运行的一些临时文件。

（6） webapps 目录，存放 Tomcat 服务器自带的 2 个 Web 应用 host-manager 应用和 manager 应用。ROOT 子目录下存放默认首页，即输入 http://127. 0. 0. 1:8080/后启动的页面。

（7） work 目录，存放 JSP 文件在运行时被编译成的二进制文件（Servlet）。在 localhost 文件夹下包含多个子文件夹，其中第一个文件夹“_”对应 Web 服务的根，Tomcat 执行主 Web 应用的 jsp 页面时生成的临时文件，将存储在“\Tomcat 6. 0\work\Catalina\localhost_”文件夹中。其他文件夹分别对应虚拟目录，每建立一个虚拟目录，在 localhost 文件夹中将创建一个同名的子文件夹。用户可以删除整个 localhost 子文件夹，来删除所有的临时文件。

有时修改页面内容后，仍然显示修改以前的内容，这时可以尝试把 work/Catalina/localhost 目录中所有内容删除。如果删除时出现无法删除提示，需要关闭 Tomcat，然后再删除。重启 Tomcat 即可正确显示修改后预期的内容。

在 Tomcat 6. 0 以前的 Tomcat 5. 5 中，有三个不同的 lib 目录，分别存储在/server、/common 和/shared 目录下，这些 lib 目录都可以放 jar 文件。它们的区别主要在于：

（1） /server/lib 目录下的 jar 文件只可被 Tomcat 服务器访问；

（2） /common/lib 目录下的 jar 文件可以被 Tomcat 服务器和所有 Web 应用访问；

（3） /shared/lib 目录下的 jar 文件可被所有 Web 应用访问，而不能被 Tomcat 服务器访问。

在用户自己的站点中，WEB-INF 目录下也可以建 lib 子目录，在 lib 子目录下也可以放各种 jar 文件，但这些 jar 文件只能被当前 Web 应用访问。

接下来即可进行 Tomcat 的配置，分成四个方面：根据上述目录结构，进行相应的环境变量设置，配置 Tomcat 服务端口，设置 Tomcat 服务根目录，建立虚拟目录。

2. 配置 Tomcat 环境变量

Tomcat 为 JSP 的容器，要在 Windows 下运行 JSP，需要安装 Java 开发环境，同时需要一些特殊的环境设置，包括以下四个系统环境变量，具体内容应根据安装路径设置。

（1） 添加 Tomcat 主目录环境变量：

```
TOMCAT_HOME = C: \Tomcat 6.0
```

(2) 添加 CATALINA_HOME 环境变量:

```
CATALINA_HOME = C:\Tomcat 6.0
```

(3) 更新 CLASSPATH 环境变量:

```
CLASSPATH = .; %JAVA_HOME%\lib; %TOMCAT_HOME%\lib
```

(4) 更新 PATH 环境变量:

```
PATH = %PATH%; %TOMCAT_HOME%; %TOMCAT_HOME%\bin
```

上述环境变量的配置和 Java 环境变量的配置方法相同。设置完成后,重新启动计算机,使设置生效,然后再启动 Tomcat。

需要特别注意的是,如果该步骤的环境变量配置不对或者 server.xml 文件配置不对(见下面的介绍),Tomcat 将无法启动。另外,如果 Web 应用中只是一般的 htm 文件,不配置环境变量,网站也可以浏览,因此,Tomcat 启动后,并不意味着所有的需要运行用户 Web 的设置都完成或正确。

在实际应用中,一般需要更改三个基本配置:修改服务端口、修改网站的根路径和建立虚拟目录。在以前的 Tomcat 版本中,这些配置比较复杂。在 Tomcat 6.0 中,这些配置都是通过 Tomcat 主配置文件 conf/server.xml 完成的。

3. 修改服务端口

在 Tomcat 的安装过程中,可以设置 Tomcat 服务端口,默认值为 8080。安装完成后,如果需要修改服务端口,可通过 Tomcat 主目录下的 conf 目录中的 server.xml 文件完成。不同的 Tomcat 版本,主配置文件 Server.xml 的内容不同。

对于 Tomcat 6.0.x,利用 UltraEdit 或其他文本编辑器打开\Tomcat 6.0\conf\目录下的 server.xml 文件,定位元素<Connector port="8080">,可以看到 Tomcat 服务的设置端口为 8080,如图 2-40 所示。

```
<!-- A "Connector" represents an endpoint by which requests are received
     and responses are returned. Documentation at :
     Java HTTP Connector: /docs/config/http.html (blocking & non-blocking)
     Java AJP  Connector: /docs/config/ajp.html
     APR (HTTP/AJP) Connector: /docs/apr.html
     Define a non-SSL HTTP/1.1 Connector on port 8080
-->
<Connector port="8080" protocol="HTTP/1.1"
           connectionTimeout="20000"
           redirectPort="8443" />
```

图 2-40 Tomcat 的服务端口信息

修改 Web 服务端口为 http 的默认端口 80。注意,如果是在 Windows 平台中,并且安装了 IIS,则修改的端口号不要和 IIS 中的 Web 服务冲突。修改完毕后,保存该文件,然后重启 Tomcat 服务器,这样 Tomcat 就在新的端口提供服务了。

4. 修改网站根路径

不同的 Tomcat 版本,设置 Web 应用根的方法也不相同。在 Tomcat 5.5.x 中,修改网

站根路径的方法有两种：一种是修改 C:\Tomcat 5.5\conf\目录下的 Tomcat 主配置文件 server.xml，一种是建立 ROOT.xml 文件。在 Tomcat 6.0.x 中，设置 Tomcat 根的方法非常简单，只需要修改 Tomcat 主配置文件 conf/server.xml 即可。

用记事本打开 Tomcat 主配置文件 server.xml，定位到文档尾部的＜Host＞元素，添加一个上下文元素(＜Context＞)，来设置 Tomcat 的根。例如，如果将 d:\GSL3.0 设置为 Tomcat 的根，设置如图 2-41 所示。

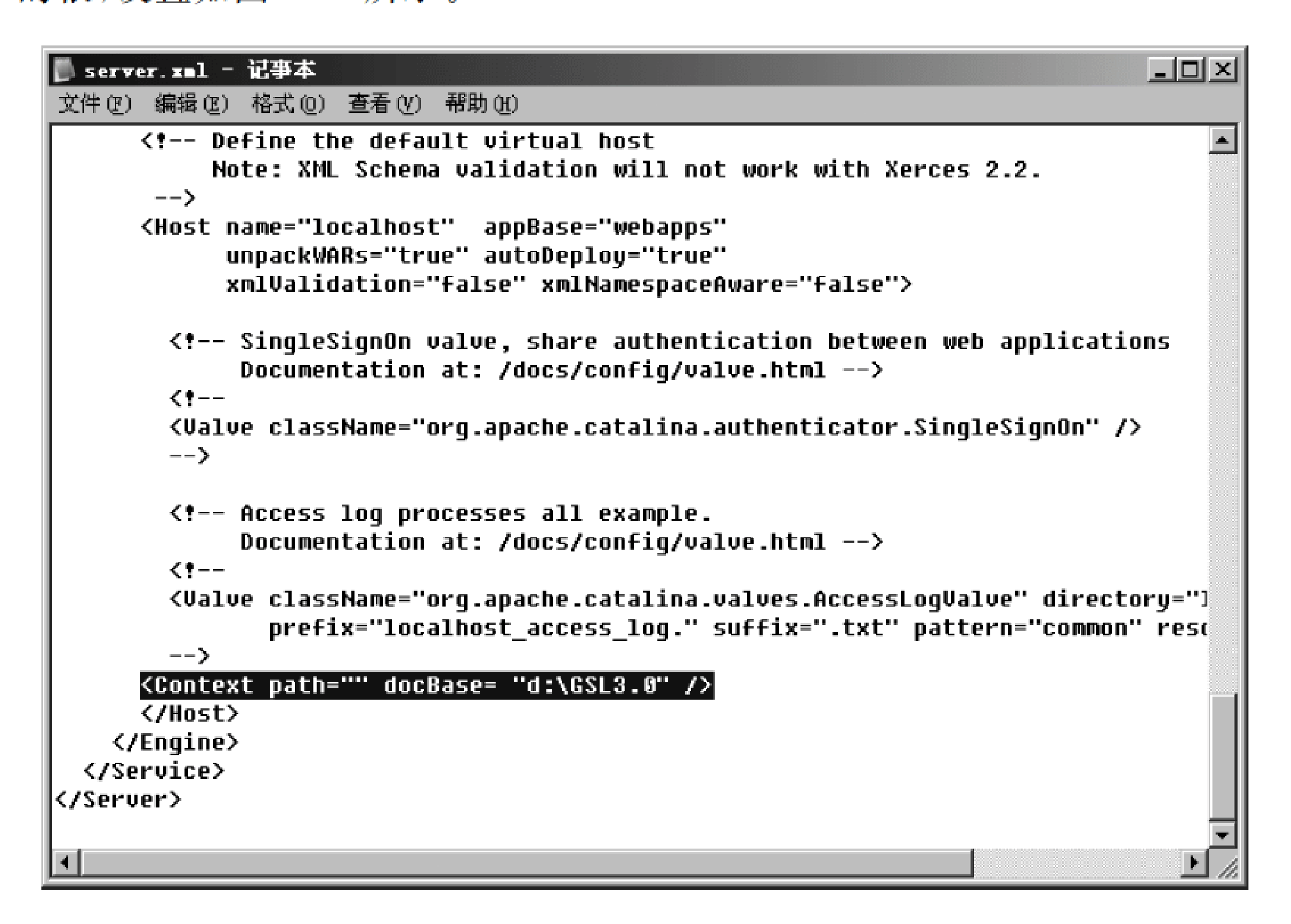


图 2-41　设置 Tomcat 服务的根

需要特别注意的是，Tomcat 区分大小写，＜Context＞元素的第一个字母一定为大写，且文件夹名称大小写也必须和实际一致。修改完毕后，在任务栏上停止 Tomcat，再重新启动，打开浏览器，将运行根中的 index.jsp 程序页面。

如果还要建立虚拟目录，只需要再增加不同的＜Context＞元素即可，详细介绍见 2.5.7 节。在 server.xml 中，可以设置多个不同的虚拟目录。

5. 设置 Web 应用首页

在 Windows IIS 中，可以设置一个 Web 站点的首页(即：登录一个站点，在不指定下载文件时默认的下载文档，一般是存储在站点主目录下的 index 文件)。在 Tomcat 中，如何设置站点首页呢？

在 Tomcat 中，站点首页是通过 web.xml 文件完成的，web.xml 文件又称为站点配置文件。在每一个 Web 应用中，往往在主目录下包含一个 WEB-INF 子目录，其中存储了该站点的配置文件 web.xml。此外，在 Tomcat 的 conf 文件夹下也包含一个 web.xml 文件，内容如下：

```
1166 -  <welcome - file - list>
1167        <welcome - file> index.html </welcome - file>
```

```
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

</web-app>
```

Tomcat 的 conf/web.xml 文件是对所有 Web 应用的一个公共配置。对于一个具体的 Web 应用,如果包含自己的 WEB-INF/web.xml 文件,当两个配置冲突时,则自己的 web.xml 配置将覆盖 conf/web.xml 中的设置。一般情况下,只需要修改 conf/web.xml 配置文件即可,不需要单独设置每一个应用的 WEB-INF/web.xml 文件。

2.5.6 建立并部署 Web 应用

在默认情况下,Tomcat 指向一个默认的 Web 应用(\Tomcat 6.0\webapps\ROOT),在 webapps 文件夹下,还包含其他的几个 Web 应用,如 jsp-examples、servelets-examples 等。

下面介绍在 Tomcat 下新建 Web 应用的方法和步骤。

1. 规划 Web 应用目录结构

对于一个 Web 应用,包含了大量的网页文件,为了更好地管理和维护。应该按照一定的规则组织文件。常用的方法是按照 Web 站点功能建立文件夹,分别存储相应的页面文件。图 2-42 是我们根据一个常用的 Web 应用规划的文件夹结构。

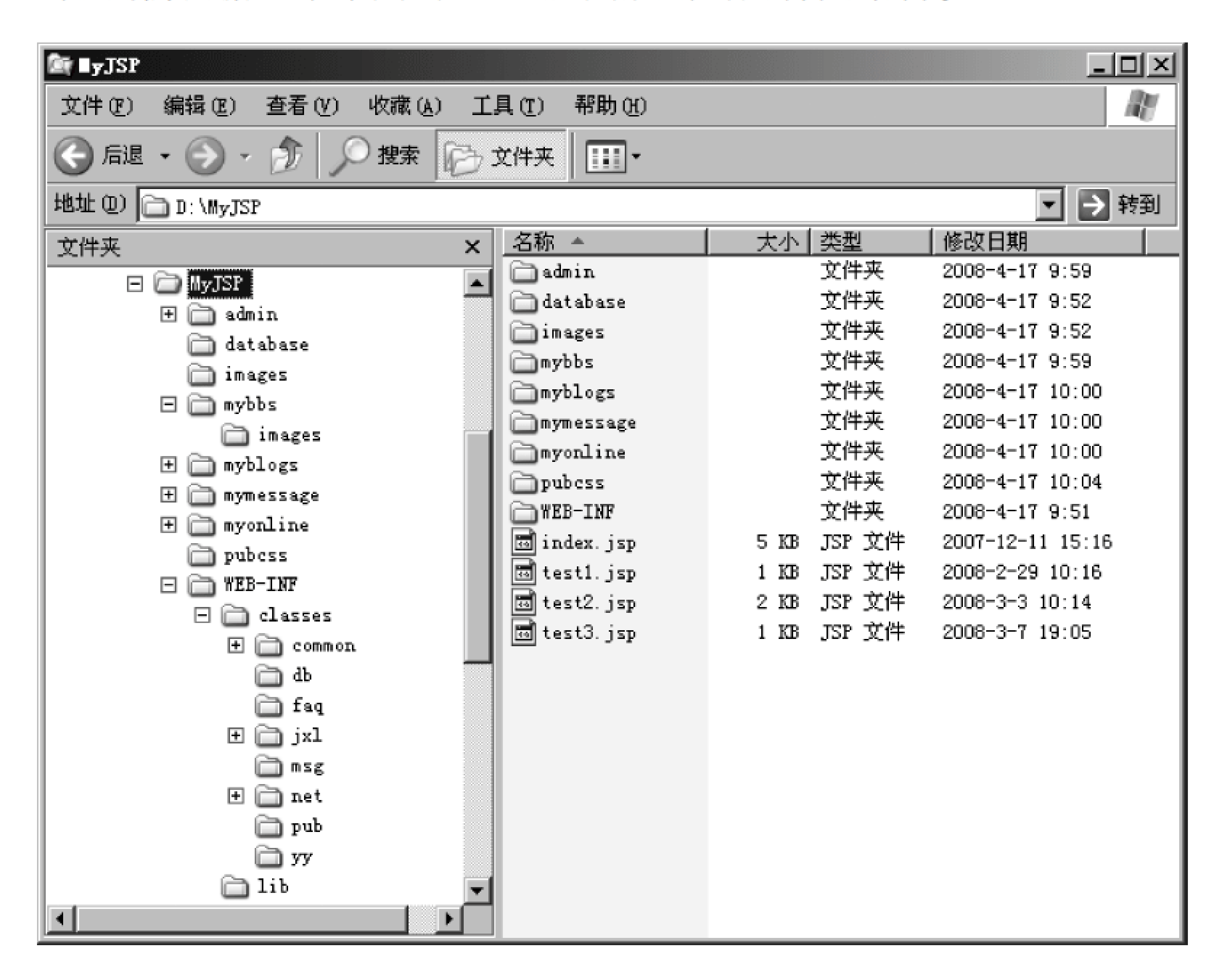


图 2-42 Web 应用目录结构示例

将每一类功能相关的页面、图片组织到一个文件夹中,例如 mybbs、myblogs、myonline 等,在这些文件夹中,还可以定义子文件夹,例如定义 images 文件夹,存储所用到的图片。

在站点主目录下，通常还可以定义 database 文件夹，存储站点数据库文件。定义 pubcss 文件夹，存储用户定义的样式表。

当然，还有一个 WEB-INF 文件夹，存储 Web 应用的配置文件 web.xml 以及定义 classes 和 lib 两个子文件夹，存储 Web 中用户定义的类。用户定义的大量的 JavaBean 都是存储在 WEB-INF/classes 文件夹中的，里面通常还定义不同的包，即子文件夹。

在站点主目录中，包含了 Web 应用的首页文件 index.jsp，也可以包含一些其他的常用文件，这些文件通常是公用的，不便于保存到一个具体的功能文件夹中。

2. WEB-INF 目录

在 Tomcat 中，每一个 Web 应用，主目录下往往都包含一个 WEB-INF 目录，用于放置一些配置文件与不希望外部程序访问的隐私文件，在网络上是不允许访问该文件夹的。在 WEB-INF 目录下有一个 Web 应用部署文件 web.xml，对当前应用程序进行相关设置，如设置 Web 应用的默认首页文件等。

在 WEB-INF 目录下还可以创建 classes 和 lib 子目录。classes 目录用于放置 Web 应用程序所需调用的类，如 JavaBean。在运行过程中，Tomcat 类装载器先装载 classes 目录下的类，再装载 lib 目录下的类。如果两个目录下存在同名的类，classes 目录下的类具有优先权。lib 目录主要是放置需要引入的 jar 文件，应用程序导入的包先从这里开始寻找，其次到容器的全局路径下 $TOMCAT_HOME/lib 下寻找。

3. Web 应用配置文件 web.xml

对 Web 应用的配置是通过 Web 应用配置文件 web.xml 实现的，类似于 Windows IIS 中的站点属性对话框的配置。在 Tomcat/conf 下包含一个 Web 应用配置文件 web.xml，它是所有 Web 应用的公共配置文件。此外，在每一个 Web 应用中，在主目录下的 WEB-INF 子目录中，都包含一个 web.xml 文件，它是该 Web 应用的部署文件。当两个配置中的项目冲突时，则自己的 web.xml 配置将覆盖 conf/web.xml 中的设置。

在 web.xml 配置文件中，根元素是<web-app>，其中定义了站点的各种配置，主要包括以下几个方面。

（1）网站名称和说明。包括三个 xml 元素，分别是<description>、<display-name>、<icon>，用于设置站点的描述、显示名称和图标。例如，Tomcat 自带的 manger 应用的 web.xml 中的站点说明 xml 元素内容如下：

```
<display-name>Tomcat Manager Application</display-name>
<description>
  A scriptable management web application for the Tomcat Web Server;
  Manager lets you view, load/unload/etc particular web applications.
</description>
```

（2）Servlet 的名称和映射。servlet-mapping 元素包含两个子元素 servlet-name 和 url-pattern，用来定义 servlet 所对应的 URL。

（3）Session 的设定。session-config 包含一个子元素 session-timeout，用于定义 Web 站点中的 session 参数。例如，设定会话时间为 20 分钟，对应的 xml 元素内容为：

```
<session-config>
    <session-timeout>20</session-timeout>
</session-config>
```

(4) mime 映射。mime-mapping 包含两个子元素 extension 和 mime-type,定义某一个扩展名和某一 MIME Type[①] 映射。例如:要在 Tomcat 中打开 excel 文件,需要在 web.xml 中做如下设置:

```
<mime-mapping>
    <extension>xls</extension>
    <mime-type>application/vnd.ms-excel</mime-type>
</mime-mapping>
<mime-mapping>
    <extension>csv</extension>
    <mime-type>application/vnd.ms-excel</mime-type>
</mime-mapping>
```

(5) 错误处理。error-page 元素包含三个子元素 error-code、exception-type 和 location,将错误代码(Error Code)或异常(Exception)的种类对应到 Web 站点的相应页面。例如:

```
<error-page>
    <error-code>404</error-code>
    <location>/error404.jsp</location>
</error-page>
```

(6) 默认首页设置。对应的元素声明一般形式为:

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

一般情况下,只需要配置 Tomcat 的 conf/web.xml 公共配置文件即可,不需要为每一个 Web 应用配置其 WEB-INF/web.xml 文件。

4. 修改 Tomcat 配置

在测试 Web 应用之前,需要对 Tomcat 做相应的设置,使得 Tomcat 指向用户的 Web 应用(例如 D:\MyJSP),修改如下。

(1) 修改 Tomcat 主配置文件\Tomcat 6.0\conf\server.xml,设置 Web 服务的端口号为 80,同时,修改默认 Tomcat 服务的根,在 server.xml 的尾部,添加下列元素(图 2-40):

```
<Context path="" docBase="d:\MyJSP" />
```

(2) 设置站点首页,可以修改 Tomcat 配置文件\Tomcat 6.0\conf\web.xml,设置 Web

---

① MIME 类型就是设定某种扩展名的文件用一种应用程序来打开的方式类型,当该扩展名文件被访问的时候,浏览器会自动使用指定应用程序来打开。多用于指定一些客户端自定义的文件名,以及一些媒体文件打开方式。

应用的一些常用配置，默认首页为 index.jsp，无须修改。

需要注意的是，如果已经启动了 Apache Server，首先应该在 Windows 的“开始”菜单中，在“程序”组中找到 Apache HTTP Server 程序组，执行 Stop 命令，停止 Apache Server。

5. 测试新的 Web 应用

当上述修改完毕后，在任务栏中右击 Tomcat 图标，选择 Shutdown：Tomcat 命令，关闭 Tomcat。然后在“开始”菜单中重新启动 Tomcat，尝试运行用户 Web 应用。

在站点根下，建立一个简单的站点首页文件 index.jsp，代码如下：

```
<%@ page contentType="text/html;charset=gb2312" %>
<html>
<head>
<title>Hello,JSP</title>
</head>
<body>
<p align="center"><% out.println("你好, JSP...!"); %></p>
<%
      String datestr = "" ;
      java.util.Date now = new java.util.Date() ;
      java.text.DateFormat df = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm") ;
      datestr = df.format(now) ;
%>
现在的时间是：<% =datestr %>
</body>
</html>
```

打开 IE 浏览器，输入 http://127.0.0.1:8080/1.jsp，显示如图 2-43 所示。

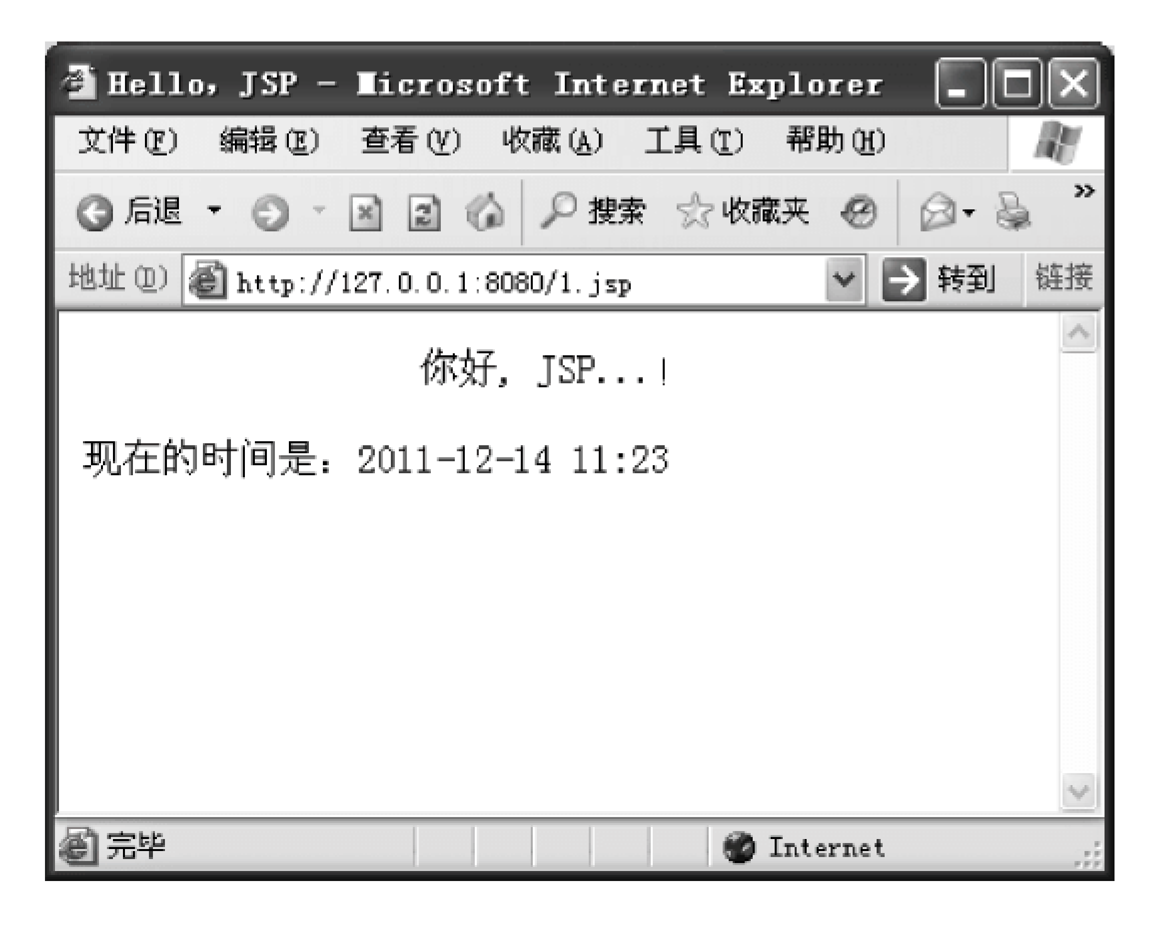


图 2-43　第一个 JSP Web 应用首页

表明 Tomcat 已经运行了用户的 Web 应用 D:\MyJSP 目录下的首页文件 index.jsp。用户可以在主目录下创建其他的 jsp 文件，在 IE 的地址栏中输入 http://127.0.0.1:8080/文件名(包含扩展名)即可执行相应的 jsp 文件。

如果 jsp 文件中含有 Java 脚本程序，必须要保证 Tomcat 和 J2SDK 的环境变量设置正

确,否则 Web 应用将不能运行。如果 Web 页是.htm 文件,运行与环境变量的配置无关。

在应用中,如果遇到不能打开 Web 应用首页文件 index.jsp,应检查该 Web 应用主目录下的 WEB-INF\web.xml 配置文件,同时检查 Tomcat 下的公共配置文件 conf\web.xml,确认两者是否配置一致。如果修改了某个 jsp 页面,但重新运行仍显示原先内容,需要删除 Tomcat\work\Catlinia\localhost 中的所有临时文件,也可以直接删除 localhost 子文件夹。

2.5.7 Apache 和 Tomcat 的整合

通过上一节的介绍,我们已经非常清楚 Apache 和 Tomcat 的功能,即 Apache 是一个 Web 服务器,Tomcat 为应用服务器。虽然 Tomcat 内置了一个 Apache 的 HTTP 服务,但是它仅仅对 JSP 程序体现出比较好的执行效率和性能,对于静态页面的处理速度远不如 Apache。

可见,为了提高 Web 系统的整体性能,需要将 Apache 和 Tomcat 进行整合配置。假设,系统已经成功安装了 Apache(端口号为 80)和 Tomcat 服务器(端口号为 8080),未作特殊配置,要将两者进行有机整合,可以分为以下三个方面。

在示例站点 d:/haosite 中,新建一个 index.jsp 首页文件,代码同上。修改 Apache 主配置文件 httpd.conf,设置 Apache 服务器的首页为 index.jsp。操作如下。

用记事本打开 Apache 主配置文件 httpd.conf,找到首页配置指令段:

```
<IfModule dir_module>
    DirectoryIndex index.html index.jsp
</IfModule>
```

将默认的 index.html 改为 index.jsp。

重新启动 Apache,打开浏览器,在地址栏中输入 http://127.0.0.1/,页面中没有显示当前时间,可见,服务器页中的脚本程序未执行。

然后,在浏览器地址栏中输入 http://127.0.0.1:8080/,则能够正确显示页面(图 2-43),可见,Tomcat 能够提供 Web 服务,同时,Tomcat 执行了服务器页中的脚本程序。

整合 Apache 2.2 和 Tomcat 6.0 就是要保证 Apache 执行 Web 服务,并实时调用 Tomcat 执行服务器页面中的脚本程序。要进行 Apache 和 Tomcat 的整合,可以采用三种方法。

(1) 利用 Apache 自带的 mod_proxy 模块使用代理技术来连接 Tomcat。http_proxy 模式是基于 HTTP 的代理,因此它要求 Tomcat 必须提供 HTTP 服务,也就是说必须启用 Tomcat 的 HTTP Connector。

打开 Apache 配置文件 httpd.conf 文件,找到如下代码:

```
#LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_http_module modules/mod_proxy_http.so
```

将注释去掉,即可启用 proxy_module 和 proxy_http_module。

然后,在 Apache 的 httpd.conf 文件中增加如下两行:

```
ProxyPass / http://localhost:8080/
ProxyPassReverse / http://localhost:8080/
```

重启 Apache,就可通过 Apache 来访问 Tomcat 的服务了。

(2) ajp_proxy 连接方式。跟 http_proxy 方式一样,都是由 mod_proxy 所提供的功能。配置也是一样,只需要把 http://换成 ajp://,同时连接的是 Tomcat 的 AJP Connector 所在的端口。配置如下。

在 Apache 中修改 httpd.conf 文件,找到如下代码:

```
#LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_ajp_module modules/mod_proxy_ajp.so
```

将注释去掉,启用 proxy_module 和 proxy_ajp_module。

在 Apache 的 httpd.conf 文件中增加以下几行代码。

```
#禁止使用 proxy_ajp 代理的目录
ProxyPass /examples/ !
#使用 proxy_ajp 代理;下面的配置,是把所有目录全用代理(当然,还会跟上面的禁用配置组合成
完整的规则)
ProxyPass / ajp://127.0.0.1:8009/
ProxyPassReverse / ajp://127.0.0.1:8009/
```

重启 Apache,就可通过 Apache 来访问 Tomcat 的服务了。

(3) 通过 JK 模块整合,配置需要安装 JK,修改 Apache 服务器的配置文件 httpd.conf,加载 JK 模块以及指定 JK 配置文件信息。

① workers.properties:到 Tomcat 服务器的连接定义文件。

② uriworkermap.proprties:URI 映射文件,用来指定哪些 URL 由 Tomcat 处理,也可以直接在 httpd.conf 中配置这些 URI,但是独立这些配置的好处是 JK 模块会定期更新该文件的内容,使得修改配置时无须重新启动 Apache 服务器。

相对于上面的方法(1)和方法(2),该方法配置麻烦,但效率较高,详细介绍略。

采用上述任何一种方法,配置完成后,在浏览器地址栏中输入 http://127.0.0.1/,显示结果页面和 http://127.0.0.1:8080/相同,则表明整合成功,此时 Apache 调用了 Tomcat,执行服务器页面中的 jsp 服务器脚本程序。

2.6 虚拟主机与虚拟目录

在 Web 站点的配置中,虚拟目录和虚拟主机是两种常见的配置,无论是 Windows Server /IIS 还是 Apache/Tomcat,都可以配置虚拟目录和虚拟主机。虽然其配置方法不尽相同,但其基本概念是一样的,本节将以 Apache/Tomcat 为例,介绍虚拟目录、虚拟主机的概念及其配置方法。

2.6.1 虚拟主机及其配置

在同一台 Web 服务器上运行多个网站的技术称为虚拟主机。使用虚拟主机,可以让多个站点共享同一台物理机器,减少系统的运行成本,并且可以减少管理的难度。此外,对于个人用户,也可以使用这种虚拟主机方式来建立有自己独立域名的 WWW 服务器。

1. 虚拟主机的类型

虚拟主机分为基于 IP 的虚拟主机和基于域名的虚拟主机两种形式。

1) 基于 IP 地址的虚拟主机

所谓基于 IP 地址的虚拟主机方式,是指不同的主机名解析到不同的 IP 地址,提供虚拟主机服务的机器上同时设置有这些 IP 地址。在具体实现中,如果每个虚拟主机(网站)使用不同的端口号,则每台虚拟主机运行一个 Apache 服务。如果多个虚拟主机使用相同的端口号、不同的主机名,则它们将共享一个 Apache 服务。

2) 基于域名的虚拟主机

在 HTTP 1.1 中增加了对基于主机名的虚拟主机的支持。具体说,当客户程序向 WWW 服务器发出请求时,客户想要访问的主机名也通过请求头中的"Host:"语句传递给 WWW 服务器。WWW 服务器程序接收到这个请求后,可以通过检查"Host:"语句,来判定客户程序请求的是哪个虚拟主机的服务,然后再进一步地处理。

基于域名的虚拟主机相对比较简单,因为只需要配置你的 DNS 服务器将每个主机名映射到正确的 IP 地址,然后配置 Apache HTTP 服务器,令其辨识不同的主机名就可以了。基于域名的服务器也可以缓解 IP 地址不足的问题。如果没有特殊原因必须使用基于 IP 的虚拟主机,最好还是使用基于域名的虚拟主机。

但是,基于域名的虚拟主机配置有其不足,因为,在该类配置中,多个网站共享一个 IP 地址和端口,多个虚拟主机使用一个 Apache 服务程序,各个虚拟主机共享同一份 Apache,因此有 CGI 程序运行时,安全性不高。

2. 虚拟主机的设置

设有两个公司共享一台 Web 服务器,公司域名分别是 www.company1.com 和 www.company2.com。两公司在 DNS 域名注册时均设定这台 Web 服务器的 IP 地址。该 Web 服务器采用基于名字的虚拟主机设置。

为测试方便,使用本地 DNS 解析机制,在 Windows\system32\drivers\etc\hosts 文件,添加上述的域名解析,IP 均为 127.0.0.1。

在 Apache2 安装目录下的\conf\extra\文件中,找到 httpd-vhosts.conf 文件,进行如下配置。

(1) 为每个虚拟主机建立<VirtualHost>段:

```
# ServerName 是网站域名,需要跟 DNS 指向的域名一致
# DocumentRoot 是网站文件存放的根目录
<VirtualHost *:80>
    ServerName www.company1.com
    DocumentRoot "D:/company1"
</VirtualHost>
```

如果想在现有的 Web 服务器上增加虚拟主机,必须也为现存的主机建造一个<VirtualHost>定义块。这个虚拟主机中 ServerName 和 DocumentRoot 所包含的内容应该与全局的 ServerName 和 DocumentRoot 保持一致。还要把这个虚拟主机放在配置文件

的最前面，来让它扮演默认主机的角色。

(2) 打开 httpd.conf 文件，开启虚拟主机配置文件：

```
# Virtual hosts
Include conf/extra/httpd-vhosts.conf
```

(3) 重启 Apache 服务，访问虚拟主机。

现在 Web 服务器上有三个站点，即：中心主机(Mainhost)，在 httpd.conf 中设置的 DocumentRoot "D:/haosite"；虚拟主机 www.company1.com(在\conf\extra\httpd-vhosts.conf 配置)；虚拟主机 www.company2.com(在\conf\extra\httpd-vhosts.conf 配置)。

然后就可以通过域名 www.company1.com 和 www.company2.com 访问 company1 和 company2 两个虚拟主机了。如果没有进行 Apache 和 Tomcat 的整合，将不运行 JSP 中的服务端脚本程序，但能够返回网页。

3. Apache 与 Tomcat 虚拟主机一致

在站点的实际运行中，需要 Apache 和 Tomcat 的融合，因此只配置 Apache 的虚拟主机，并不能保证整个融合的有效运行。完整的配置总结如下。

(1) 在 Apache 的 httpd.conf 配置中加载需要的代理模块，取消下面两行的注释：

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

如果在尾部包含下述指令，将其注释掉：

```
#ProxyPass / http://localhost:8080/
#ProxyPassReverse / http://localhost:8080/
```

(2) 修改 httpd-vhosts.conf，添加虚拟主机，并集成 Tomcat：

```
NameVirtualHost *:80
<VirtualHost *:80>
    ServerName www.company1.com
    ProxyPass / http:// www.company1.com:8080/
    ProxyPassReverse / http:// www.company1.com:8080/
</VirtualHost>
NameVirtualHost *:80
<VirtualHost *:80>
    ServerName www.company1.com
    ProxyPass / http:// www.company2.com:8080/
    ProxyPassReverse / http:// www.company2.com:8080/
</VirtualHost>
```

(3) 打开 httpd.conf 文件，开启虚拟主机配置文件：

```
# Virtual hosts
Include conf/extra/httpd-vhosts.conf
```

(4) 在 Tomcat 6.0 的 server.xml 中定义上述同名的虚拟主机。

删除原先的<host>定义，添加新的虚拟主机定义，即在尾部添加两个<Host>定义，

如图 2-44 所示。

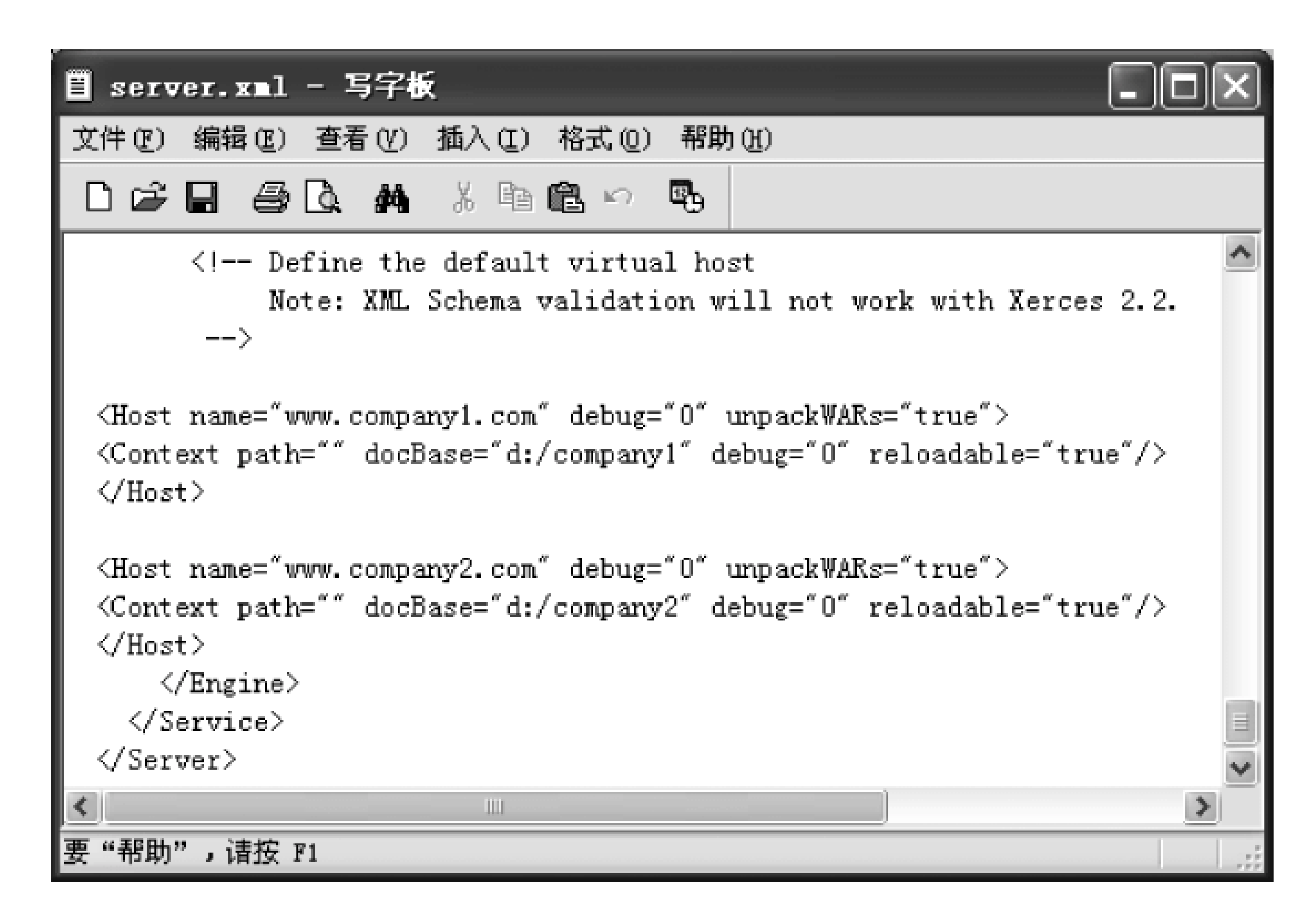


图 2-44 Tomcat 虚拟主机设置

当上述配置结束后,就可以正确地访问虚拟主机,并执行 Tomcat 了。例如在浏览器地址栏中输入 http://www.company1.com/,显示页面如图 2-45 所示。



图 2-45 访问 Apache 虚拟主机

2.6.2 虚拟目录及其配置

在 Web 站点中,站点内容是由站点主目录下的所有子文件夹和文件构成的。所谓"虚拟目录"就是一个指向站点主目录以外的物理目录的指针,理论上讲,该文件夹不属于站点的一部分,但为了通过站点访问到该文件夹的内容,就在站点根下创建一个指向该外部文件夹的指针,即虚拟目录,以便于在 URL 中书写路径,定位其中的文件。

设置了虚拟目录后,访问该 Web 站点主目录外的文件,通过使用虚拟目录即可定位,在浏览器地址栏中输入 http://127.0.0.1/虚拟目录/文件名。

1. Tomcat 中虚拟目录的设置

在 Tomcat 6.0 中,使用虚拟目录,非常简单,只需要修改 Tomcat 主配置文件 conf\

server.xml，在尾部增加一个新的<Context>元素即可。例如，建立一个到 d:/haosite 的虚拟目录，在 server.xml 中，在根目录设置的后面(图 2-41)，添加下述内容：

```
<Context path = "/hao" docBase = "d:\haosite" reloadable = "true" crossContext = "true"
        Debug = "0" workdir = "d:\haosite\work">
</Context>
```

其中，path="/hao"定义了根下的一个虚拟目录 hao，docBase="d:\haosite"为虚拟目录 hao 对应的物理路径。参数 reloadable 设置为 true，表明修改 Servlet 文件、jsp 文件后，不用重启 Tomcat 即可生效。

保存 server.xml 文件，然后重启 Tomcat 服务器，可以在地址栏中通过虚拟目录访问 d:/haosite 中的网页文件了，例如 http://127.0.0.1/hao/index_hao.jsp。

此时，在 Tomcat 的临时文件夹 C:\Tomcat 6.0\work\Catalina\localhost 中，自动创建一个与虚拟目录同名的临时文件夹 hao，存储该虚拟目录生成的临时文件。

2. Apache 中虚拟目录的设置

在 Apache 的配置文件 httpd.conf 下搜索 Directory，得到 Apache 虚拟目录例子。记着开启虚拟主机模块。Apache 配置虚拟目录分三种情况。

(1) 如果未配置虚拟主机，在 httpd.conf 中，建一个虚拟目录 elearning，对应的物理目录为 d:/hao/elearning。配置如下：

```
Alias /elearning "D:\hao\elearning"
<Directory "D:\hao\elearning">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
```

(2) 如果 Apache 配置了虚拟主机，可以将上述指令添加到 httpd-vhosts.conf 虚拟主机的声明中，即：

```
<VirtualHost * :80>
   ServerName www.company1.com
   DocumentRoot "D:/company1"
   ProxyPass / http://www.company1.com:8080/
   ProxyPassReverse / http://www.company1.com:8080/

   Alias /elearning "D:\hao\elearning"
   <Directory "D:\hao\elearning">
      AllowOverride None
      Options None
      Order allow,deny
      Allow from all
   </Directory>
</VirtualHost>
```

(3) 如果 Apache 已经和 Tomcat 进行了整合，还需要在 Tomcat 的 Server.xml 同时为

虚拟主机添加虚拟目录,配置如下:

```
<Host name="www.company1.com" debug="0" unpackWARs="true">
   <Context path="" docBase="d:/company1" debug="0" reloadable="true"/>
   <Context path="/elearning" docBase="D:\hao\elearning" debug="0" reloadable="true"/>
</Host>
```

当虚拟主机/虚拟目录配置完成后,运行 Web 浏览器,在地址栏中输入一个 URL,访问虚拟目录下的页面,例如 http://www.company1.com/elearning/1.jsp,可以看到页面正确显示。

2.7 Web 服务器的远程管理与维护

随着互联网的发展,服务器的远程管理成为最主要的管理模式。在 Windows Server 2003 平台中,只要在服务器上进行相应的配置,就可以实现对服务器的远程管理。

2.7.1 使用终端服务和远程桌面

在 Windows Server 中,都提供了终端服务组件。因此,管理员可以在 Windows 服务器上安装 Windows 服务组件“终端服务”和“远程桌面 Web 连接”,然后管理员就可以使用 Windows XP 中的“远程桌面连接”(在“附件”→“通信”程序组中),来登录到 Web 服务器,实现对服务器的操作。

选择“附件”→“通信”→“远程桌面连接”,首先打开“远程桌面连接”对话框,如图 2-46 所示。



图 2-46 “远程桌面连接”对话框

输入要连接的计算机的 IP 地址,单击“连接”按钮,打开“Windows 登录”对话框;输入远程计算机上的一个本地账户和密码,即可登录到远程服务器,显示其桌面,接下来就可以如在本地一样对远程的计算机进行操作和管理了。

2.7.2 基于浏览器的服务器远程管理

在 Windows Server 平台中,在 IIS 中,在“万维网服务”组件中,包含了“远程管理(HTML)”子组件,安装该组件后,可以对 Windows 服务器进行远程管理。

具体设置如下。

(1) 在“计算机管理”控制台中,在“服务和应用程序”节点下,展开“Internet 信息服务”

管理单元。

(2) 在需要远程管理的 Web 站点上右击,打开快捷菜单,执行“属性”命令,打开 Web 站点属性对话框。在 Web 站点选项卡中,记下该站点的 TCP 端口号。

(3) 在 Web 站点属性对话框中,选择“目录安全性”选项卡,在“IP 地址和域名限制”选项组中,单击“编辑”按钮,打开“IP 地址和域名限制”对话框,执行下列操作之一。

- 如果要允许所有计算机远程管理 IIS,选择“授权访问”单选按钮。
- 选择“拒绝访问”单选按钮,然后单击“添加”按钮,打开“授权访问”对话框;选择要授权访问的“单机”、“一组计算机”或者“域名”,按照系统提示进行操作。

当 Web 服务器上启用了基于浏览器的 Internet 服务管理器 (HTML)后,就可以使用基于浏览器的 Internet 服务管理器了 。

在浏览器地址栏中输入 https://Web 服务器网址(域名或 IP 地址):8098/,按 Enter 键,显示“连接到”对话框;输入一个管理员权限的用户账户和密码,则打开“服务管理”站点,即通过 Web 接口远程维护 Windows Server 2003 服务器界面,如图 2-47 所示。

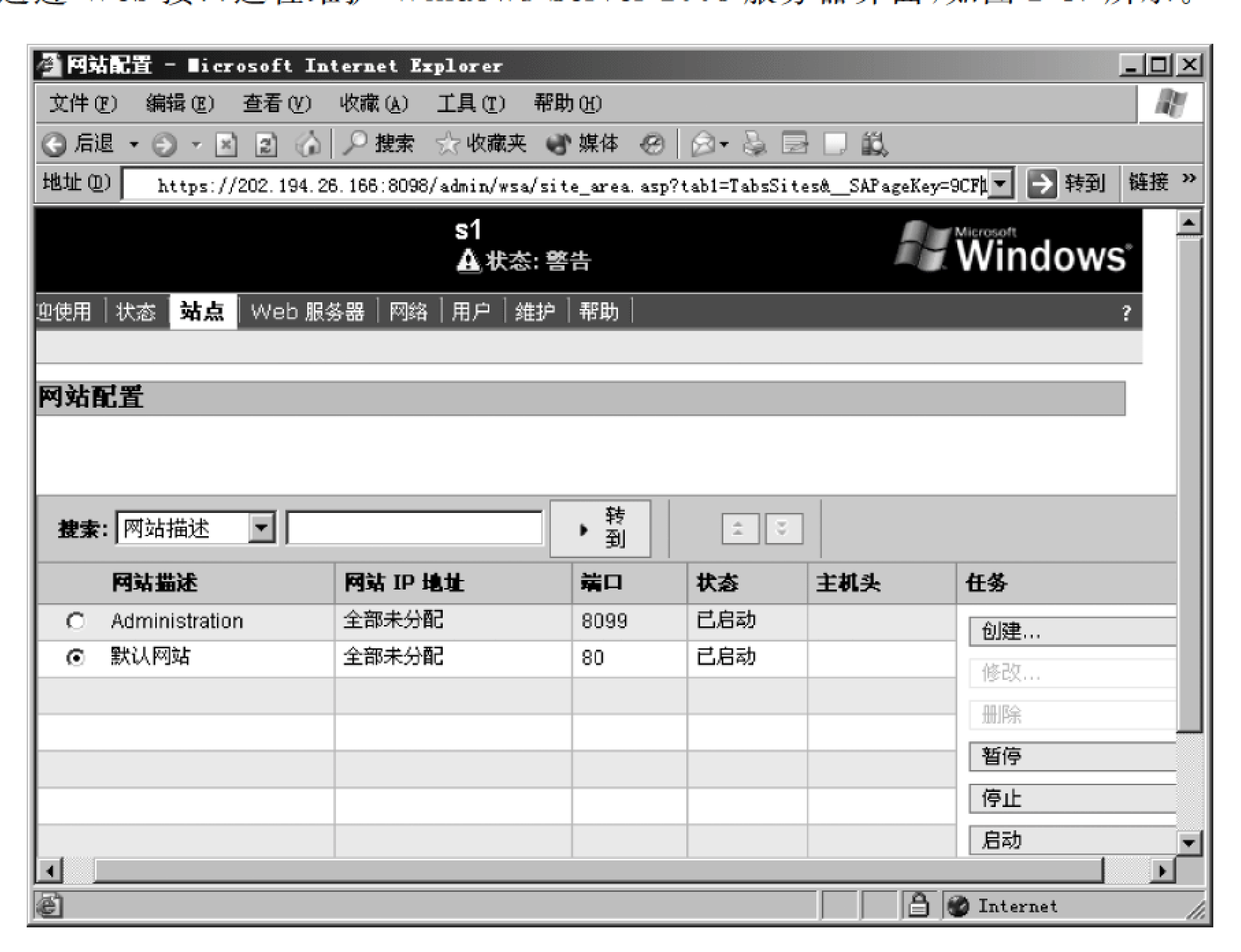


图 2-47 通过 Web 接口远程维护 Windows Server 2003 服务器界面

通过 Web 接口,可以实现 Windows Server 2003 服务器的远程维护,包括站点、Web 服务器、网络、用户等维护功能。

2.7.3 网站内容的远程维护

如果要对 Windows 服务器平台中的网站进行远程管理,可以有很多方式,除了上述几种管理方式外,还可以利用 FTP 和 FrontPage 等进行远程管理,其中采用 FTP 管理网站具有更好的通用性,可以管理各种类型的网站;但是,使用 FrontPage 只能管理 IIS 中的网站,

如果是 Tomcat 中的网站,则不能通过 FrontPage 来管理。

使用 FTP 管理网站,需要对 Web 服务器进行如下配置。

(1) 在 Web 服务器计算机上搭建一个 FTP 站点,设置其主目录为要管理的网站主目录。

(2) 用户登录 ftp 站点,即可看到网站主目录,接下来就可以对网站进行维护操作了。例如:修改网页,首先将要修改的网页下载,修改完毕后再上传,覆盖原先的网页。

可见,该方式与网站是 IIS 的,和 Tomcat 无关,也不需要对要管理的网站的属性进行特殊的设置,只需要 FTP 指向站点主目录即可。

对网站的远程维护还可以通过 FrontPage、Dreamweaver 等工具来完成,这些可视化的网页制作工具,目前都包含站点远程维护功能。其中,对于要使用 FrontPage 管理的网站,不仅要在服务器端添加 FrontPage 扩展,还必须在 IIS 中对网站属性进行设置,可见 FrontPage 只能远程维护 IIS 中的网站。

本章小结

本章首先介绍了操作系统、Web 服务器和 Web 应用开发的关系,然后以 Windows Server/IIS 为例,讲解了 Web 服务器的安装和配置过程,介绍了其中的概念,包括端口号、主目录、默认文档以及安全性等。在此基础上,重点讲解了 Apache 服务器和 Tomcat 服务器的功能、安装和配置。讲解了 Apache 服务器和 Tomcat 服务器的关系,Tomcat 服务器与 JSP、Java 之间的关系以及 Java 运行环境的安装和环境变量配置。介绍了虚拟主机和虚拟目录的概念,讲解了 Apache 和 Tomcat 中虚拟主机和虚拟目录的配置方法。最后,对 Web 站点的远程管理和内容维护进行了介绍。

习题 2

1. 什么是 Web 服务器?有哪些主流的 Web 服务器产品?
2. 什么是 IIS? IIS 6.0 包括哪些可选组件?简述它们的功能。
3. 简述在 IIS 中 Web 站点的创建过程。
4. 在 Windows Server/IIS 中,当连接新建的 Web 站点时出现如图 2-48 所示的“输入网络密码”对话框,为什么?如何解决?

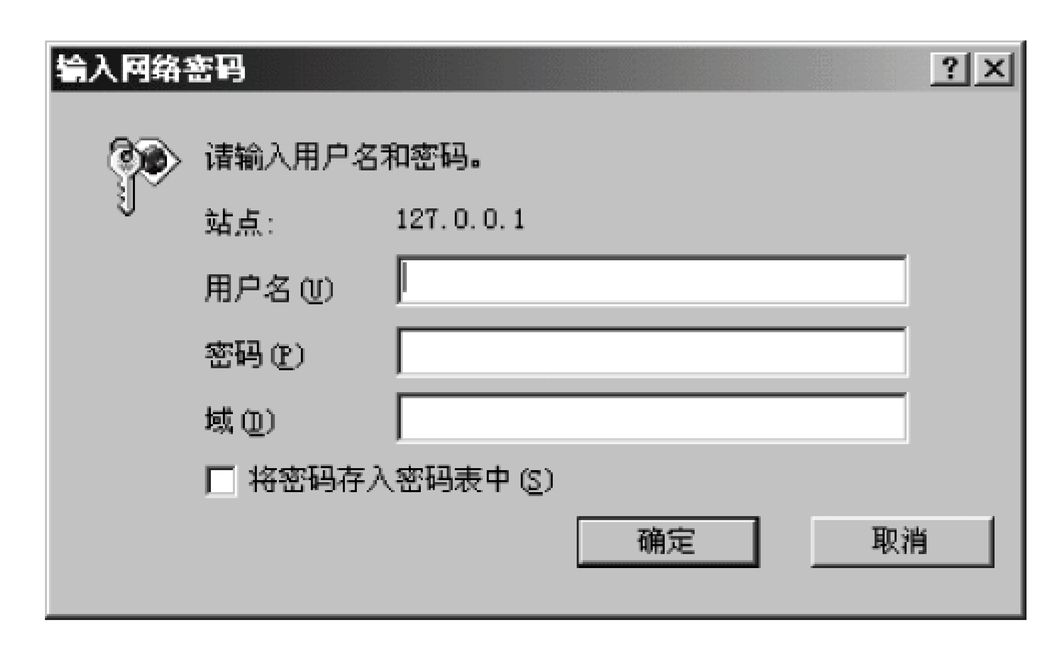


图 2-48 “输入网络密码”对话框

5. 当连接到一个 IIS 中的 Web 站点，浏览一个 ASP 页时，显示“网页无法显示”提示页面，并且在页面中提示“您试图从目录中执行 CGI、ISAPI 或其他可执行程序，但该目录不允许执行程序”，为什么？如何解决？

6. 简述 Apache 服务器和 Web 服务器的功能。

7. Java 运行环境包括哪些内容？安装 JDK 后，需要设置哪些系统环境变量？简述设置每个环境变量的目的。

8. 安装一次 Tomcat 6.0，简述 Tomcat 目录结构中各个目录的功能。

9. 在 Web 应用中，简述 WEB-INF 文件夹的作用，Web 应用配置文件 web. xml 的功能是什么？

10. 什么是虚拟主机和虚拟目录？虚拟主机有几种方式？

11. 在一台 Windows 服务器上，能同时安装 IIS 和 Apache/Tomcat 吗？如果能，在 IIS 下创建的网站和 Tomcat 下的网站能同时运行吗？如何配置？

12. 要实现对 Web 站点的远程管理，有哪些方式？说明每种方式的特点。

第3章 HTML和XML基础

【本章导读】

在Web中，超文本标记语言(HTML)或可扩展标记语言是W3C发布的用于进行网页书写的标准和规范。虽然都称为标记语言，但两者的功能完全不同，HTML规范是一种数据展示技术，它定义了一组固定的标记，来规范网页内容在浏览器中的显示形式。XML规范则是一种数据表达技术，它的重点是解决数据的逻辑结构和语意表达，以实现文档的结构化和数据处理的自动化。目前，XML文档数据的显示通常还要转换为HTML格式来显示。

本章将从广义标记语言的概念出发，介绍标记语言的概念和功能，对HTML规范进行总结，并结合Web中一些典型的网页，介绍HTML规范中的主要标记的功能及用法。在XML相关技术方面，重点讲解XML元语言规范、文档类型定义和XML Schema技术的思想和出发点，同时讲解XML扩展样式语言，以及简要介绍XML路径语言XPath，使大家对这些技术规范的功能及相互关系有一个比较深入的了解，为Web应用和开发打下坚实的基础。

【知识要点】

3.1节：标记语言的概念、标准通用标记语言(SGML)、超文本标记语言(HTML)、可扩展标记语言(XML)。

3.2节：超文本标记语言(HTML)文档结构，标记，标记属性，段落、字体标记，图片标记，超链接标记，表格标记，表单标记，输入域标记，CSS技术，样式表，CSS选择符，图层标记，区段标记，帧标记，浮动帧，脚本语言标记。

3.3节：可扩展标记语言(XML)、XML文档结构、文档声明、处理指令、XML解析器。

3.4节：文档类型定义(DTD)、元素、元素声明、基本元素、复杂元素、属性、属性声明、内置属性类型、实体、实体定义、外部dtd文件。

3.5节：XML Schema(模式)、命名空间、简单数据类型、简单数据类型定义、复杂数据类型、复杂数据类型定义、元素声明、属性声明、W3C CSD内置数据类型、预定义元素、模式文件。

3.6节：XML文档对象模型(DOM)、可扩展样式语言(XSL)、模板、XML路径语言XPath、路径表达式、路径步、谓词、XPath轴、XPath函数。

3.7节：XML Spy集成开发环境、dtd文档的创建、模式文档的创建、XML实例文档的创建、系统建模、有效性验证。

3.1　标记语言及其发展

广义上的标记语言可以理解为对内容进行描述的规范或标准。例如，在出版印刷行业，编辑人员在进行文档内容编辑时对内容所做的标记，通过这些标记符号来表达对内容的修改信息。在 Web 中，超文本标记语言则通过标记来定义内容在浏览器中的显示样式。随着语义 Web 的发展以及异构环境下的数据交换需求，可扩展标记语言则主要用于对数据结构和数据内容进行标记，成为重要的数据表达、交换和集成标准，对数据的显示则通过其他相关语言来完成，实现了数据内容和显示的分离。

3.1.1　标准通用标记语言

标准通用标记语言(SGML)是一个用来定义在电子表格中如何对文件的结构和内容进行描述的国际标准(ISO-8879)。时间可以追溯到 1969 年，当时美国 IBM 公司的研究人员开始设计一种名为 GML(Generalized Markup Language，通用标记语言)的语言，在印刷、统计等需要大规模数据处理的行业和部门的支持下，这项研究工作持续了十几年，于 1980 年推出了 SGML，并于 1986 年获得国际标准化组织(ISO)的批准。其后，SGML 的发展较为平稳，并不为其领域之外的人们所广泛了解。直至 1991 年，当超文本标记语言(HTML)问世之后，人们才开始认识 SGML。

为了满足各种不同的页面表达需要，SGML 设计得非常复杂，SGML 的正式规范达 500 多页，因此使用起来很不方便，使得它未能得到普及和大规模的应用。虽然 SGML 没有被广泛应用，但是 SGML 定义了标记语言的基本概念，奠定了标记语言发展的技术基础。

现在，在 Web 中普遍应用的 HTML 和 XML 都是在 SGML 的基础上开发成功的，可以说它们都是 SGML 的一个子集。作为互联网信息共享的技术规范，标记语言对互联网的发展起到了巨大的推动作用。

3.1.2　超文本标记语言

超文本标记语言(HTML)起源于标准通用标记语言(SGML)，由世界上最大的粒子物理研究实验室欧洲核子研究中心(CERN)于 1991 年首先提出，是推动 Web 迅速发展的原动力，被誉为互联网发展历史上的第一个里程碑。

在互联网发展的早期，为了在各种网络环境之间、不同文件格式之间进行交流，在 SGML 基础上，CERN 提出了超文本标记语言(HTML)的概念。HTML 是一种用来制作超文本文档的简单标记语言，它定义了一组标记符号(tag)，对文件的内容进行标注，指出内容的输出格式，如字体大小、颜色、背景颜色、表格形式、各部分之间逻辑上的组织等，从而实现了文件格式的标准化。简单地说，HTML 文件包含了文档数据和显示样式两部分，其中文档数据是显示在 Web 浏览器中的数据内容，显示样式则规定了这些内容在浏览器中以何种格式、样子呈现给用户。通过统一使用支持 html 的浏览软件，用户可以在任意异构的网络环境中，阅读同一个文件，得到相同的显示结果，并可以对文件进行跳跃式阅读，展现了很强的表现力。

HTML 主要版本和发布时间如下。

(1) HTML 2.0，Internet 工程任务组中的 HTML 工作组开发完成了 HTML 2.0，于 1996 年发布。

(2) HTML 3.2，W3C 于 1997 年 1 月 14 日将其列为推荐版本，在 HTML 2.0 标准中添加了诸如字体、表格、Java 程序、浮动、上标、下标等特征。

(3) HTML 4.0，W3C 于 1997 年 12 月 18 日将其列为推荐版本，第二个稍作修正的 HTML 4.0 版本于 1998 年 12 月 24 日发布。HTML 4.0 中最重要的特征是引入了样式表(CSS)技术，使网站样式与内容分离，使得网站结构更加清晰，内容更加简洁。

(4) HTML 4.01，W3C 于 1999 年 12 月 24 日将其列为推荐版本，是 HTML 4.0 的升级版本，它对原版本做出了部分修正。

(5) HTML 5.0，作为下一代超文本标记语言的标准，草案最早于 2004 年提出，目前仍处于开发阶段，W3C 组织预测，其完成时间预计要等到 2012 年以后或更长的时间。

在下一代 HTML 标准 HTML 5.0 的发展过程中，W3C 组织希望净化 XHTML 2.0，回归第一版 HTML 的设计理念。但是，这样的设计理念遭到了 W3C 之外的一些重要的 HTML 专家，包括浏览器厂商、Web 开发人员、作者和其他有关人员的质疑，2004 年，他们成立了一个独立的工作组，即 WHATWG(Web 超文本应用程序技术工作组，Web Hypertext Application Technology Working Group)，为新的 HTML 版本提出了新的设计方向。

3.1.3 可扩展 HTML

在 HTML 的发展过程中，暴露出一些影响其发展的缺陷，例如：HTML 的标记固定，HTML 只是一种表现技术，不能表达语义；不能适应现在越来越多的网络设备和应用的需要，比如手机、PDA、信息家电都不能直接显示 HTML；由于 HTML 代码不规范、臃肿，浏览器需要足够智能和庞大才能够正确显示 HTML；数据与表现混杂，页面要改变显示，就必须重新制作 HTML。为此，W3C 不再继续开发 HTML。

2000 年底，W3C 制定了可扩展 HTML，即 XHTML，它是 HTML 向 XML 过渡的一个桥梁。2000 年 1 月 20 日发布 XHTML 1.0，XHTML 1.0 是 HTML 4.01 基于 XML 的形式。2002 年 8 月 5 日，发布 XHTML 2.0 的第一个工作草案，其最大的特点就是取消了向后兼容性，去除了原先版本中的一些标记。例如，不再支持使用很少的<em>(强调)和<strong>标记等，这就使得原先的一些网页在 XHTML 2.0 规范的浏览器中不能正确显示。

需要说明的是，在 XHTML 的研发过程中，不确定的东西还很多，作为 HTML 向 XML 的一种过渡技术，唯一确定的就是它要更好地表达文档中的语义和结构，将文档的内容和表现技术能够更好地分离，更好地实现互联网中的数据交换和展示。

3.1.4 可扩展标记语言

HTML 的不足推动了 XML 的产生和发展，其核心思想是实现数据和显示的分离。1998 年 2 月 10 日，XML 工作组正式向 W3C 提交了 XML 的最终推荐标准，这就是 XML

1.0 标准。XML 规范定义了标记语言的主要特征，例如 DTD、XML Schema 等基本要素，这些要素可以很好地用于定义数据，实现异构环境下的数据交换。

对 XML 文档内容的显示、查询及操作则通过其他一系列的规范来实现，这些相关的规范包括可扩展样式语言(XSL)、XML 路径语言(XPath)、XML 查询语言(XQuery)、可扩展连接语言(XLL)以及 XML 文档对象模型(DOM)与简单应用程序接口(SAX)等，通过这些规范来实现对 XML 文档的显示及其他各种操作。

3.2　超文本标记语言

超文本标记语言(HTML)是在 SGML 基础上发展起来的，是互联网中应用最为广泛的标记语言，被称为 World Wide Web 的通用出版语言。在互联网中，绝大多数的网页都是通过 HTML 标记语言书写的，所有的 Web 浏览器都支持 HTML 规范，并能很好地显示 HTML 网页文件。即使是 XML 文档，也通常需要通过 XSLT 转化为 HTML 格式进行显示。

3.2.1　HTML 标记语法和文档结构

HTML 文档是纯文本文件，由“显示内容”和“标记”两部分组成。标记描述显示内容以何种形式在浏览器中显示，也就是说“标记(Tag)”是对内容的标记。标记封装在由小于号(<)和大于号(>)构成的一对尖括号之中，标记一般分首标记和尾标记，它们成对出现。首标记用于开启某种形式的显示，尾标记用于关闭首标记开启的显示功能。例如：<u>欢迎光临</u>，首标记<u>开启下划线功能，尾标记</u>关闭下划线功能。该语句在浏览器中将把文本串“欢迎光临”加上下划线显示。

1. 标记类型

标记分为“单标记”和“双标记”两种类型。“单标记”是指只需单独使用就能完整地表达意思的一类标记，单标记不标记任何内容。这类标记的语法是：

```
<标记>
```

常用的单标记有换行标记
、水平线标记<hr>等。

另一类标记称为“双标记”，由“首标记”和“尾标记”两部分构成，必须成对使用。首标记告诉 Web 浏览器从此处开始执行该标记所表示的功能，而尾标记告诉 Web 浏览器在这里结束该功能。首标记名称前加一个斜杠(/)即成为尾标记。这类标记的语法是：

```
<标记>标记内容</标记>
```

其中“标记内容”部分就是要被这对标记施加作用的部分。例如，如果需要标记一段文本要红色显示，则将文本放在双标记…中即可，即：

```
<font color = "#FF0000">您好</font>
```

标记可以被 Web 浏览器解释，对所标记的内容以特定的样式在浏览器中显示。对于其他的文本阅读器，例如记事本等，则不能解析标记的含义。

2. 标记属性

在 HTML 规范中,标记都设定了默认的显示样式,此外,标记通常还有相应的属性。标记属性分为一般属性和事件属性两种类型,一般属性对应一个相应的属性值,事件属性则对应一段程序代码,当标记上的对应事件发生时,激活相应的事件对应的程序。

设置属性标记的一般形式是:

```
<标记 属性="属性值|程序" 属性="属性值|程序" …>
```

各属性之间无先后次序,属性之间用空格分开。属性也可省略(即取默认值),属性值两侧为西文双引号("),可以使用西文单引号('),或省略不写。

例如:单标记<hr>表示在文档当前位置画一条水平线,默认情况下是从窗口中当前行的最左端一直画到最右端。另外,<hr>标记还有 size、align、width 等属性,其中 size 属性定义线的粗细,默认为 1;width 属性定义线的长度,可取相对值(整个窗口的百分比),也可取绝对值(屏幕像素点的个数),默认值是 100%。align 属性设置对齐方式,可取 left(左对齐,默认值)、center(居中)和 right(右对齐),要画一条 300 像素宽、居中显示的水平线,可写作<hr width="300" align="center">。

再如,在图片标记<img>中可设置事件属性,写为<img src="images/logo.jpg" onclick="window.alert('hi')">,其中 onclick 就是事件属性,当在图片上单击时,事件被激活,打开一提示窗口。

3. 文档结构

一个 HTML 文档以<html>标记开始,以</html>标记结束,表示这对标记间的内容是 HTML 文档。HTML 文档分成文件头和文件体两个部分,由相应的标记来区分。HTML 文档总体结构如下:

```
<html>
<head>
        头部信息
</head>
<body>
        文档主体
        (语句部分)
</body>
</html>
```

其中,<head>…</head>之间是文件头,由一系列子标记构成,如定义文档标题的<title>标记等,若不需头部信息则可省略此标记。<body>…</body>之间是文件体,表示正文内容的开始,<body>标记一般不能省略。

3.2.2 文件头标记及子标记

在 HTML 文档中,<head>…</head>标记对之间的部分称为文件头。根据 Web 的工作原理,在 Web 服务器和 Web 浏览器的 HTTP 通信中,HTTP 头为浏览器和服务器提供辅助信息,这些辅助信息也可以写在 HTML 文档的头部,为浏览器、搜索引擎等提供信息。例如:设置网页内容字符编码、网页关键字等,这可以使浏览器按照设定的字符编码正

确地显示网页内容，以及让搜索引擎记录网页关键字。

文件头中主要的标记有<title>、<meta>、<script>等，下面分别介绍。

1. <title></title>标记

<title></title>标记出现在<head>…</head>标记对内，用于标识网页主题，其中的内容将在浏览器的标题栏中显示，不出现在页面内。

2. <meta>标记

meta 即“元”的意思，meta data 即元数据，是关于数据的数据。元标记<meta>是最重要的辅助性标记，往往不引起用户的注意，但是它对于网页是否能够被搜索引擎检索、提高网页在搜索列表的排序起着关键的作用，是一个非常有价值的标记。

<meta>标记为单标记，没有尾标记。<meta>标记共有两个属性，分别是 http-equiv 属性和 name 属性，不同的属性又有不同的参数值，这些不同的参数值实现了不同的网页功能。

1）http-equiv 属性

http-equiv 相当于 HTTP 头，向浏览器传回一些有用的信息，以帮助正确显示网页内容，与之对应的属性值为 content，content 中的内容其实就是各个参数的变量值。meat 标记的 http-equiv 属性语法格式是：

```
<meta http-equiv="参数" content="参数变量值">
```

其中 http-equiv 属性主要有以下几种参数。

（1）content-type，设定页面文档类型及使用的字符集。

例如：<meta http-equiv="content-type" content="text/html; charset=GB 2312">，该元标记告知浏览器，文档为 HTML 文档，使用的字符集为 GB 2312。如果是 XML 文档，可以设置 content 属性值为 content="text/xml; charset=GB 2312"。

（2）expires，用于设定网页的到期时间。一旦网页过期，必须到服务器上重新下载。

例如：

```
<meta http-equiv="expires" content="Thur, 8 May 2008 18:18:18 GMT">
```

（3）pragma，禁止浏览器从本地计算机的缓存中访问页面内容。

例如：

```
<meta http-equiv="pragma" content="no-cache">
```

该种设定访问者将无法使用脱机浏览功能。

（4）refresh，自动刷新并指向新页面。

例如：

```
<meta http-equiv="refresh" content="60; url=new.htm">
```

则浏览器将在 60 秒后，自动转到 new. htm。利用该功能，可以显示一个封面提示页面，在若干时间后，再自动转移到其他页面。

如果不设置 URL 项，浏览器则刷新本页，这就实现了 Web 聊天室定期刷新的特性。

(5) window-target,强制页面在当前窗口以独立页面显示。

例如:

```
<meta http-equiv="window-target" content="_top">,
```

可以用来防止别人在框架中调用自己的页面。

2) name 属性

name 属性主要用于描述网页,与之对应的属性值为 content,content 中的内容主要是便于搜索引擎查找信息和分类信息用的。meat 标记的 name 属性语法格式是:

```
<meta name="参数" content="具体的参数值">
```

其中 name 属性主要有以下几种参数值。

(1) keywords,设置网页的关键字,用来告诉搜索引擎该网页的关键字是什么。

例如:

```
<meta name="keywords" content="E-learning,ontology">
```

(2) description,description 用来告诉搜索引擎网站的主要内容。

例如:

```
<meta name="description" content="This page is about E-learning etc.">
```

(3) author,标注网页的作者。

例如:

```
<meta name="author" content="brion@mail.abc.com">
```

(4) robots,告诉搜索机器人需要索引的页面有哪些。content 的参数有 all、none、index、noindex、follow、nofollow,默认值为 all。

3. <link>标记

<link>标记定义了文档之间的包含。在 HTML 的头部可以包含任意数量的<link>标记,<link>标记带有很多属性,下面是一些常用的属性。

(1) type,用于指定被包含的文件类型。例如,text/css 是指包含一个层叠样式表文件。

(2) rel,定义 HTML 文档和所要包含资源之间的链接关系,可能的取值很多,最为常用的取值是 stylesheet,用于包含一个固定首选样式表单。

(3) href,指向被包含资源的 URL 地址。

例如,如果文档包含一个外部的 css 文件,在文档头部应该定义如下:

```
<link type="text/css" rel="stylesheet" href="mystyle.css">
```

4. <base>标记

<base>标记定义了文档的基础 URL 地址,在文档中所有的相对地址形式的 URL 都是相对于这里定义的 URL 而言的。一篇文档中的<base>标记不能多于一个,必须放于头部,并且应该在任何包含 URL 地址的语句之前。

<base>标记包含如下属性。

1) href 属性

href 属性,必选属性,指定了文档的基础 URL 地址。例如,如果希望将文档的基础 URL 定义为 http://www.abc.com,则可以使用如下语句:

```
<base href = "http://www.abc.com">
```

当定义了基础 URL 地址之后,文档中所有引用的 URL 地址都从该基础 URL 地址开始,例如,对于上面的语句,如果文档中一个超级链接指向 gsl/welcome.htm,则它实际上指向的是如下 URL 地址:http://www.abc.com/gsl/welcome.htm。

2) target 属性

target 属性同框架一起使用,它定义了当文档中的链接被单击后,在哪一个框架中打开页面。如果文档中超级链接没有明确指定打开页面的目标框架,则使用这里定义的地址代替。常用的 target 的属性值有:

(1) _blank,表明在新窗口中打开链接指向的页面。

(2) _self,在当前文档的框架中打开页面。

(3) _parent,在当前文档的父窗口中打开页面。

(4) _top,在链接所在的完整窗口中打开页面。

例如:<base target="_blank"> 表明页面上所有的链接都在新窗口打开。

5. 背景音乐标记<bgsound>

在文档头内,还可以定义背景音乐,标记为<bgsound>,用以插入背景音乐,可插入的音频文件类型有 Wave 文件(*.wav)、Midi 序列文件(*.midi)、Real Audio 文件(*.ram、*.ra)、AIFF 声音文件(*.aif、*.aifc、*.aiff)、AU 声音文件(*.au、*.snd)。

<bgsound>标记为单标记,一般形式是:

```
<bgsound src = " " autostart = "" loop = "">
```

标记的属性有:

(1) src 属性,给出音乐文件的 url 值。

(2) autostart 属性,设置音乐文件播送结束后的处理,如果为 true,则自动播放音乐,为 false 则结束不播放,默认值为 false。

(3) loop 属性,设置是否自动反复播放,loop=2 表示重复两次,infinite 表示重复多次。

3.2.3 文档体标记及其属性

在<body>…</body>标记对之间的部分称为 html 文档的文档体。文档体中描述的是浏览器窗口中显示的内容。无论网页多复杂,它们都是由文本、图片、超链接等内容构成的,这些页面内容都由相应的 HTML 标记来标记,它们都是<body>标记的子标记。

在讲解具体文档内容标记以前,先介绍文档体标记<body>。<body>标记是一个非常重要的标记,含有大量的属性,许多重要的网页功能都是通过<body>标记的属性实现的。

1. 一般属性

<body>标记的一般属性用于页面的一般性设置,见表 3-1。

表 3-1 <body>标记一般属性

属性	用途	举例
bgcolor="#rrggbb"	设置页面背景颜色	<body bgcolor="red">,红色背景
background	设置页面背景图片	background="images/bg1.jpg"
bgproperties	设置成 fixed,则背景图案不滚动	bgproperties=fixed
topmargin、leftmargin、bottommarigin、rightmargin	设置页面内容的上、下、左、右边距,像素值	<body leftmargin="0" topmargin="20">
text="#rrggbb"	设置文本颜色	<body text="#ff0000">,红色文本
link="#rrggbb"	设置未阅读过的超文本链接颜色,默认值是蓝色	<body link="blue">,链接为蓝色
vlink="#rrggbb"	设置阅读过的超文本链接颜色,默认值是紫色	<body vlink="#ff0000">,红色
alink="#rrggbb"	设置动作中的超文本链接颜色,默认值是紫色	<body alink="yellow">,设为黄色

颜色的设置可以通过 HTML 所给定的颜色常量名,或者 RGB(红、绿、蓝三色的组合)设置,例如"#ff0000"表示红色。各个属性可以结合使用,如:<body bgcolor="red" text="#0000ff">,设置网页的背景色为红色(red),文本为蓝色("#0000ff")。

2. 事件属性

当一个 Web 文档被加载显示或者退出(关闭),当进行移动窗口或改变文档窗口大小等操作时,会发生相应的事件,这些事件在<body>标记中通过事件属性来表达。<body>标记常见的事件属性见表 3-2。

表 3-2 <body>标记中的事件属性

事件	触发条件	事件	触发条件
onLoad	页面下载完成时触发	onDblClick	鼠标双击时触发
onUnload	退出页面时触发	onMouseDown	鼠标按键被按下时触发
onFocus	页面窗口获得焦点时触发	onKeyDown	键被按下时触发,按键的 ASCII 码值保存在 window.event.keyCode 中
onBlur	页面窗口失去焦点时触发		
onResize	窗口改变大小时触发	onKeyPress	键被按下然后被释放时触发
onScroll	单击滚动条时触发	onKeyUp	键被释放时触发
onMouseMove	鼠标移动时触发		

在上述事件中,有些事件是<body>标记特有的,有些事件可能存在于多个不同的标记中。

事件属性的值往往是一个 JavaScript 函数,来完成 Web 编程任务。在 FrontPage、Dreamweaver 等工具软件中,可以通过行为面板(在 FrontPage 2003 中,对应"格式"→"行为"菜单命令),可以显示一个标记支持的行为事件,并且可自动生成简单的行为 JavaScript 代码,从而减少用户书写程序的工作量,具体应用参考后面的章节。

【例 3-1】 HTML 标记的概念及认知。

HTML 标记就是标记内容在 Web 浏览器中以特定的格式显示,而在非 HTML 应用程序中,则不能按照标记进行显示。

在 Windows 中，用“记事本”程序输入如图 3-1 所示的 HTML 标记。

将上述内容保存为网页文档 exa3-1.htm，然后双击该文档，文档将在浏览器中打开，显示结果如图 3-2 所示。



图 3-1　用“记事本”程序创建 HTML 文档



图 3-2　HTML 文档在浏览器中显示界面

可以看到，在浏览器窗口显示红色的“您好”，对比在“记事本”中的显示，可见标记就是告知浏览器，被标记的内容要以什么方式来显示。

3.2.4　文本标记

在一个网页中，文本内容通常是网页的主要内容，在 HTML 规范中，对文本内容的标记较多，它们标记了文本内容在浏览器中的默认显示样式，要修改标记的默认显示，可以通过设置标记属性来完成。HTML 中有关文本的标记及属性见表 3-3。

表 3-3　文本标记及常用属性

分类	标　记	一般属性	事件属性
标题标记	… ，一级到六级标题标记，对应的字体逐渐渐小	id、style、class、align title	onclick、ondblclcik、onmouseover onmouseout
文本格式标记	…，字体标记	id、style、class face、size，color，设置字体、文字大小和颜色	onclick、ondblclcik onmouseover onmouseout
	…，粗体标记	id、style、class	同上
	<i>…</i>，斜体标记	同上	同上
	<u>…</u>，下划线标记	同上	同上
回车换行标记	 ，将输出位置转到下一行的开始	id、style、class	无
段落标记	… ，段落标记，标记一个段落，输出位置转到下一行开始，并增加一个空行	id、style、class align title	onclick、ondblclcik onmouseover onmouseout
走马灯标记	<marquee>…</marquee>，用于标记一行或多行滚动的文本	id、style、class height、width，设定滚动字幕的高度和宽度 bgcolor、direction scrollamount、scrolldlay	同上

续表

分类	标　　记	一 般 属 性	事 件 属 性
水平线标记	<hr>,插入一条水平线	id、style、class size、color、width	无
注释标记	<! ——注释性文字——>,用于在 HTML 文档中书写说明性文字,注释文字可以多行,内容在浏览器中不显示		

从表 3-3 可以看出,虽然不同标记的属性不完全相同,但有些属性是大多数标记所共有的,这包括:①id 属性,唯一地标识一个标记;②name 属性,为标记命名,这在 DOM 模型中对应了标记所对应的内存对象,类似于内存变量名,以便于客户端脚本程序的访问;③style 属性,设置标记的内联样式,用于修改标记的默认显示;④class 属性,设置标记的样式类,这是通过 CSS 定义的。

上述标记可以联合使用,例如:

```
<b><font face = "宋体" size = "3" color = "＃0000FF"><i><u>字体标记一</u></i></font>
</b>
```

显示效果为:

<u>字体标记一</u>

【例 3-2】 编写一段跑马灯效果的代码。

示例代码如下:

```
<marquee onmouseover = this.stop() onmouseout = this.start() scrollAmount = 1 scrollDelay = 60
  direction = up width = 150 height = 200 >
活动字幕内容第一行<br>
活动字幕内容第二行<br>
活动字幕内容第三行<br>
</marquee>
```

上述代码,将在一个区域内,垂直地滚动多行,鼠标指针指向时停止滚动,离开时继续滚动。

以下是几点说明。

(1) <marquee>标记常用的属性如下。

① align 属性,设定活动字幕的位置,取值可以是 left、center、right、top 或 bottom。

② bgcolor 属性,设定活动字幕的背景颜色,一般是十六进制数。

③ direction 属性,设定活动字幕的滚动方向,取值可以是 left、right、up 或 down。

④ behavior 属性,设定滚动的方式,主要有三种方式:behavior＝"scroll"表示由一端滚动到另一端;behavior＝"slide"表示由一端快速滑动到另一端,且不再重复;behavior＝"alternate"表示在两端之间来回滚动。

⑤ height 和 width 属性,设定滚动字幕的高度和宽度。

⑥ hspace 和 vspace 属性,设定滚动字幕的左右边框和上下边框的宽度。

⑦ scrollamount 属性，设定活动字幕的滚动距离。

⑧ scrolldelay 属性，用于设定滚动两次之间的延迟时间。

⑨ loop 属性，用于设定滚动的次数，当 loop=-1 表示一直滚动下去，直到页面更新。

默认情况下，<marquee>标记是向左滚动无限次，字幕高度是文本高度，水平滚动的宽度是当前位置的宽度；垂直滚动的高度是当前位置的高度。

（2）由于<mqrquee>标记只能作用于一段（<p>…</p>）文本，因此活动字幕为多行时，分行时只能用
标记，不能用<p>标记。

（3）字幕中也可以加入图像，代码如下：

```
<marquee><img src="image/logo.gif" width="20" height="20">欢迎光临</marquee>
```

如果希望滚动的内容带有超链接，可以将内容用<a>…标记，即<marqee>活动字幕内容</marqee>。

（4）如果是使用 FrontPage 编辑 html 文档，对于多行和插入图片的<marquee>，FrontPage 会自动地处理成一行或把<img>标记放到<marquee>…</marquee>标记的外面，此时需要通过“记事本”程序手工处理，将多行或图片放回到<marquee>…</marquee>标记内部，实现多行或者带有图片的滚动。

3.2.5 图像标记及影像地图

在一个网页中，图片是和文本一样最常见的网页内容之一。在 HTML 中，图片用两种形式出现，一种是简单的图片，另一种是影像地图。

1. 图像标记<img>

在网页中插入一幅图片，图片由<img>标记，它是一个单标记。在网页中标记一个图片的一般形式是：

```
<img src="图片的 url">
```

在<img>标记中，必须指定 src 属性，它的取值为图片的路径和文件名，为必选属性。<img>标记并不是真正地把图像嵌入到 HTML 文档中，而是将标记的 src 属性赋值为图形文件所在的路径及文件名（浏览器显示的图像格式为 gif、jpg 或 png）。这也是在 Web 服务器配置中为什么要选择保持 HTTP 连接（图 2-14），同时也是 Web 浏览器在保存页面时有多种文件保存类型的原因。下面是<img>标记的部分常用属性及其说明。

（1）src：设定图片文件的存放路径，应采用和当前页面的相对路径形式。图片文件须保存在站点主目录下的某个文件夹中。

（2）id：指定的图片 id 号，用于对图像的对象的程序访问。

（3）name：用于设定图像的名称，用于对图像对象的程序访问。

（4）height 和 width：分别用于设置图像的高度和宽度，可以与图片实际的宽度和高度不同，此时浏览器对图片进行缩放显示，可能引起图片变形。

（5）border：设置图片边框。

（6）class：设置<img>的用户自定义样式类。

(7) title：设置图像标题，当鼠标指针移到图片上时，在指针的右下角显示 title 的内容。

(8) alt：设置图像替代文字，主要用于在浏览器还没有装入图像(或关闭图像显示)时，显示此图片的替代文字。一个网页除了网页文件外，其他所包含的图片都是单独的文件，浏览器下载一个网页时，除了下载网页文件，默认情况下，会一并下载网页中包含的其他文件，例如<img>中 src 标记的文件。

(9) align：设置图像的对齐方式。除了常规的 left、right、center 外，还有 absmiddle 等取值，它将影响图片和文字混合时的对齐方式，例如在一个单元格内(<td>)，让图片和文字垂直居中。

除了上述的一般属性外，<img>标记还有大量的鼠标和键盘事件属性，例如 onload、onclick、ondbclick、onmouseover、onkeydown、onkeypress 等，在 FrontPage 等工具中，单击插入的图片，在行为面板中，可以查看图片所有的事件属性。

【例 3-3】 在一个网页中，插入一副图片 1.jpg，当鼠标指针移到图片上的时候，显示另一幅图片 2.jpg，鼠标指针移走后重新显示图片 1.jpg。

分析：根据题目要求，可以用<img>标记图片 1.jpg，然后设置<img>标记的 onmouseover 属性和 onmouseout 属性，来实现图片的切换。设网页文件名为 myimg.htm，和两个图片在同一文件夹下。示例代码如下：

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
</head>
<body>
<img src = "1.jpg" width = "200" height = "150" name = "tai"
     alt = "泰山图片" vspace = "10" hspace = "10"
     onmouseover = "tai.src = '2.jpg'" onmouseout = "tai.src = '1.jpg'" >
</body>
</html>
```

在 onmouseover 和 onmouseout 事件属性中，我们为当前的<img>标记对应的内存对象的 src 属性赋值，从而改变当前显示的图片。

2. 影像地图标记<map> </map>

所谓“影像地图”，就是在一个图片上定义一系列区域，每个区域对应一个超链接。设置影像地图，首先要通过<img>标记标记一幅图片，并设置<img>的 ismap 属性；然后再定义相应的热点区域。使用影像地图的一般形式是：

```
<img src = "图片 url" usemap = "#mapname"   ismap>
<map name = "mapname">
<area href = "url" shape = "circle" coords = "坐标值" >
<area href = "url" shape = "rect" coords = "坐标值">
…
</map>
```

其中，<map>…</map>为影像地图标记，包含一系列 area 子标记，该标记可以在图像地图中设定作用区域(又称“热点”)，每个热点区域通常对应一个超链接。热点的坐标值

由图片决定，通常需要利用 FrontPage 等工具来完成热点的定义。

3.2.6 超链接与书签

超链接是 HTML 页面的重要功能，超链接可以分为外部超链接和内部超链接两种，外部超链接是一个页面到另一个页面的链接，内部超链接则是在一个页面内书签之间的跳转。

1. 超链接标记

使用超链接标记<a>…可以定义文本超链接和图像超链接。

定义文本超链接的一般格式为：

```
<a href = "href - value" target = "target - window">文本</a>
```

如果是在图片上建立超链接，一般形式为：

```
<a href = "href - value" target = "target - window">< img src = "imge - url"></a>
```

其中，在默认情况下，当鼠标指针移到建立了超链接的文本或图片上方时，指针变成手形，表示文本或图片对应一个超链接。

下面介绍<a>标记的主要属性及功能。

（1）href 属性，设置被链接文档的 url 地址。一般形式是：

```
href = "url[ # bookmark - name][?para = value&para = value… ]"
```

其中，url 为一个网址，bookmark-name 是书签名，para 为参数，value 是参数值。它们不一地能全部出现，若是多个参数，参数名、参数值对之间用“&”分割。除了上述设置为一个完整的网址外，href 可以有多种特殊的形式，如下所示。

① href=“＃”，表示本页，这通常和 onclick 属性联合使用。

② href="＃name"，定位到当前文档的 name 书签位置。

③ href=""，定义一个空超链接，即不指向任何超链接位置。

④ href 属性为一段 javascript 代码。例如：

```
<a href = "javascript:alert('选择搜索引擎,然后单击下一步')">下一步</a>
```

（2）target 属性，定义单击超链接时打开的目标窗口。一般形式是 target="window-name"，window-name 可以取下面的常量：_self（相同框架）、_blank（新建窗口）、_top（整页）、_parent（父窗口），或者是一个存在的帧名（frame 或 iframe）。

（3）title 属性，属性值为一个字符串，鼠标指针指向超链接时，指针右下角显示标题文本。

通过 title 和 href=""空超链接属性结合，可以产生特定的效果。对尚未完成的超链接显示一个提示信息，当超链接页面完成后，再给 href 属性赋具体的 url 值。例如：

```
<a href = "" title = "is building now… ">学习论坛</a>
```

如果在提示信息中，需要换行，可以使用“&＃13;”或“&＃10;”来完成换行输出。例如：

```
title = "提示：&#13; 来宾无此权限"
```

(4) onclick 属性，接受鼠标单击，如果返回 true，则页面跳转到 href 指定的网页，否则，不执行 href 属性所定义的网页。

例如：

```
<a href = "#" onclick = "window.opener = null;window.close()">关闭</a>
<a href = "http://www.google.com" onClick = "alert('使用 Google 搜索引擎'); return(true)">
<a href = '#' onclick = "location = 'aaa.html'; return false;"> bbb </a>
```

(5) disabled 属性，开关属性，无须赋值。设置超链接显示灰化，不可用。

2. 定义书签

对于一个完整的文件，可以用它的 URL 来唯一地标识它，但对于一个文件的不同部分，怎样来标识呢？这就是书签的概念，它是通过链接标记来定义的。在文档内部可以定义书签，书签是通过<a>标记的 name 属性定义的，标识一个书签的方法为：

```
<a name = "bookmark - name">书签文本</a>
```

name 属性将放置该标记的地方标记为一个名为 bookmarkname 的书签，书签必须是一个全文唯一的标记串。有了书签后，href 属性除了指向一个网址外，还可以定位到网页内一个具体的书签，用法是 href="url#bookmark-name"，如果是同一个页内，可以写成 href="#bookmark-name"，同一网页内书签名不能重名。书签文本可以为空。

【例 3-4】 有两个网页 p1.htm 和 p2.htm，要求在 p1 中建立一个超链接以链接到 p2.htm，p2.htm 中有两个超链接，一个返回 p1.htm，一个关闭 p2.htm 页面。

下述是两个网页的代码清单。

代码清单：p1.htm

```
html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
<title>页面-1 </title>
</head>
<body>
<p><a href = "p2.htm"target = "_self">打开网页 2 </a></p>
</body>
</html>
```

代码清单：p2.htm

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
<title>页面-2 </title>
</head>
<body>
<p>
<a href = "#" onclick = "history.go(-1);return false;">返回</a>
```

```
</p>
<a href="#" onclick="window.opener=null;window.close()">关闭</a>
</body>
</html>
```

3.2.7　表格

表格(table)是网页制作中安排布局最好的工具,因为表格不但可以很好地安排文本或图像的显示位置,而且还可以任意进行背景和前景颜色的设置。在 HTML 中,表格是由<table>…</table>标记对创建的,每个<table>…</table>标记对之间包含若干<tr>…</tr>,一个<tr></tr>定义表格中的一行。每一个<tr>…</tr>标记对又包括若干个<td>…</td>标记对,每一对<td></td>标记行内的一个单元格。

此外,还可以将表格分割成三个部分,即表头、正文和脚注,分别用<thead>、<tbody>和<tfoo>来标记,它们都是由表格行构成,通过对<tbody></tbody>标记进行操作,可以控制部分行的显示与隐藏。表格标记(<table>)、行标记(<tr>)、单元格标记(<td>)、表主体标记(<tbody>)属性如表 3-4 所示。

表 3-4　表格、行及单元格等标记属性列表

标　记	常 用 属 性
<table> 表格标记	id,标识一个表格;style,设置表格样式;class,设置表格样式类
	width、height,设置表格的宽度和高度,可以是像素值,如 width="200",或窗口总宽度的百分比,如 width="80%"
	bordr,设置边框的宽度,若不设置此属性,则边框宽度默认值为 0,即无边框,无表格线;bordercolor,设置边框的颜色
	bgcolor,设置表格的背景色;background,设置背景图片
	cellspacing,设置单元格间距,即单元格之间空间的大小,默认值是 2
	cellpadding,设置衬距,即单元格内部内容之间与边框的距离
	align,设置表格的浮动对齐方式,只有 left 和 right 两种对齐方式
<tr> 行标记	hight,设置行高;width,设置行的宽度
	align、valign,设置水平和垂直对齐方式
<td> 单元格标记	id,标识一个表格;style,设置表格样式;class,设置表格样式类
	hight,设置行高;width,设置行的宽度
	align、valign,设置水平和垂直对齐方式
	colspan,设置一个单元格跨占的列数;rowspan,设置一个单元格跨占的行数
	nowrap,禁止单元格内的内容自动换行
<tbody> 表主体标记	可以将若干行定义为一个<tbody>,每个 tbody 由若干行构成。一个表格可以定义多个<tbody>,通过脚本程序可以控制它们的显示和隐藏
<thead> 表头标记	表头标记用于定义一个表格的表头,由若干行构成,使用较少
<tfoot> 表脚注标记	表脚注标记用于定义一个表格的表脚注,由若干行构成,使用较少

在 HTML 4 中,要对表格进行更加精细的设置,可以使用标记的 style 样式或 class 样式类。

【例 3-5】 定义一个高、宽为 300 像素×200 像素的 3×2 的表格,要求其在页面中水平和垂直方向居中显示。

分析:表格的水平居中容易做到,只要设置<table>的 align="center"即可,但垂直方向的居中,则没有合适的属性设置。为此,可以使用表格嵌套,首先定义一个 1×1 表格,设置其 width 和 height 属性均为 100%,然后在这个唯一的单元格内定义所要的 3×2 的表格,并设定该单元格的 align 属性为 center。示例代码如下:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<table border="0" width="100%" height="100%">
<tr>
    <td align="center">
    <table border="1" width="300" height="200" id="table1">
    <tr height="50">
        <td width="100">  </td>
        <td></td>
    </tr>
    <tr>
        <td></td>
        <td></td>
    </tr>
    </table>
    </td>
</tr>
</table>
</body>
</html>
```

在定义表格时,如果使用 FrontPage 等页面制作工具,生成的表格属性设置很乱,此时应该对代码进行手工调整,一般的设置如下。

(1) 为避免因浏览器窗口大小改变而影响表格显示,设置表格的宽度为绝对像素值,而不是比例,例如可设置 width="300",而不是"85%"。

(2) 行的高度设置应在<tr>标记中,设置 height 属性,而不要在每个单元格标记<td>中设置 height 属性。

(3) 单元格一般需设置 width 属性,如果有多个单元格,最后一个单元格不需要设置 width 属性。如果有多行,只需要在第一行设置单元格宽度,后续行将按照第一行的宽度,不需要每一行都设置单元格宽度。

【例 3-6】 使用<tbody>标记,设计一个具有标签功能的表格。即设有两个标签,当鼠标指针移动到第一个标签上时显示部分行,移动到第二个标签时,显示另外若干行。

分析:上述页面是我们在互联网上经常看到的页面功能,从题目看这是一个具有交互

功能的页面，通常需要客户端脚本程序实现。在没有学习 JavaScript 以前，我们先给出实现代码，主要是让大家体会表格中<tbody>标记的应用。

代码清单如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<style type="text/css">
.label-normal
{
    background-image:url('images/labelbgselect.gif');
    cursor: hand;
    font-size:13px;
    color: #000000;
    text-align:center
}
.label-select
{
    background-image:url('images/labelbgnormal.gif');
    cursor: hand;
    font-size:13px;
    color: #FF0000;
    font-weight:bold;
    text-align:center
}
</style>
<script language="javascript">
function selCard(n)
{
   for(i=0;i<labeltable.cells.length;i++)
       labeltable.cells[i].className="label-normal";
   labeltable.cells[n].className="label-select";
   //显示 tbody
   for(i=0;i<table1.tBodies.length;i++)
       table1.tBodies[i].style.display="none";
   table1.tBodies[n].style.display="block";
}
</script>
</head>
<body>
<table id="labeltable" border="0" cellspacing="0" cellpadding="0">
<tr height=25>
   <td class="label-select" onMouseOver="selCard(0)" width="130">排行榜</td>
   <td class="label-normal" onMouseOver="selCard(1)" width="130">贡献度</td>
   <td class="label-normal" onMouseOver="selCard(2)" width="130">其他</td>
</tr>
</table>
<table id="table1" border="0" cellspacing="0" cellpadding="0" width="100%">
<tbody>
<tr height=30>
```

```
    <td align = "center" width = "20">< img src = "images/square01.gif"></td>
    <td><a href = "../lanmu1 - news/news01.htm">教育部“十二五”本科教学工程</a></td>
</tr>
<tr height = "1">
    <td colspan = 2 background = "images/line01.gif"></td>
</tr>
<tr height = 30>
    <td align = "center">< img src = "images/square01.gif"></td>
    <td><a href = "../lanmu1 - news/news02.htm">全校核心通识课程建设项目公布</a></td>
</tr>
</tbody>
<tbody style = "display:none;">
<tr height = 30>
    <td align = "center">< img src = "images/square01.gif"></td>
    <td><a href = "../lanmu2 - news/news01.htm"> GSL5.0 系统上线预告</a></td>
</tr>
<tr>
    <td align = "center">< img src = "images/square01.gif"></td>
    <td><a href = "../lanmu2 - news/news02.htm"> GSL 过程管理问题答疑...</a></td>
</tr>
</tbody>
</table>
</body>
</html>
```

在浏览器上打开上述页面,显示结果如图 3-3 所示。

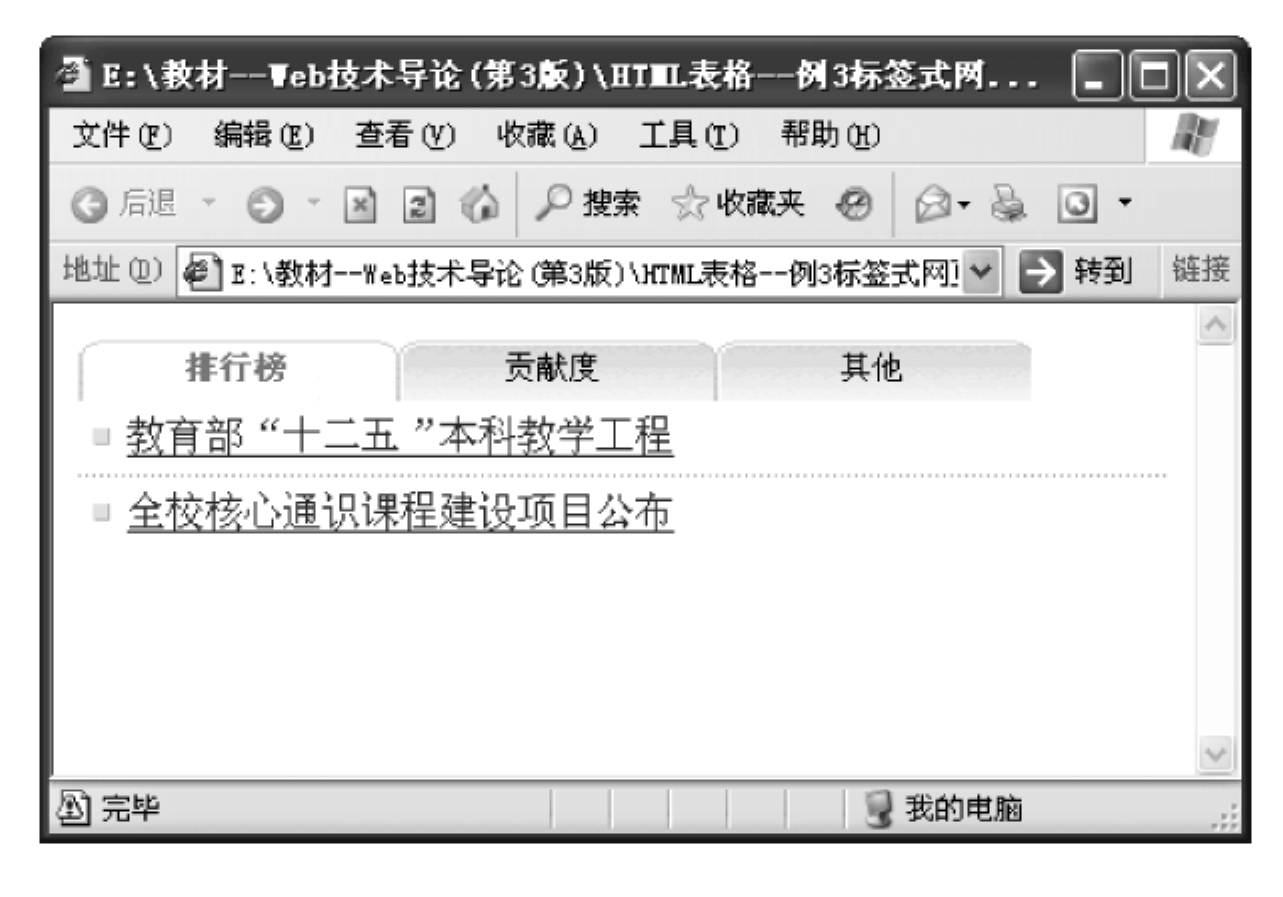


图 3-3 标签式网页的显示效果

3.2.8 表单

表单(form)在 Web 网页中用来给访问者填写信息,从而能获得用户信息,使网页具有交互能力。表单和 Windows 的对话框类似,是由若干控件组成的,用于实现和用户的交互。当用户填写完信息后做提交(submit)操作,表单的内容将从客户端的浏览器传送到 Web 服务器,由 Web 服务器中的相应的服务器脚本程序处理。

对于表单及其包含的每一个元素，浏览器在进行解释时都会在内存中创建相应的对象，这和程序设计中的变量说明是一样的，这就为用户对表单输入数据的操作提供了技术上的保证。关于表单数据的处理将在本书第 5 章和第 6 章进行讲解。本节只介绍 HTML 规范中表单及其元素的标记语法。

表单标记、常用的输入元素标记、常用属性及用法如表 3-5 所示。

表 3-5 表单标记、常用的输入元素标记、常用属性及用法

标记/输入类型	一 般 形 式	常 用 属 性
<form> 表单标记	<form name=" form-name"> … </form>	name 属性，表单名称，用于脚本编程 action 属性，设置表单处理程序的网络路径和程序名 method 属性，定义服务器表单处理程序从表单中获得信息的方式，取值为 get 或 post target 属性，设置 action 页面输出的窗口
单行文本框输入	<input type="text" …>	type 属性，设置输入域的类型 name 属性，设置输入控件的名字 value 属性，存储文本框的取值，可以设置一个初始值 size 属性，设置文本框的显示宽度 maxlength 属性，设置文本框中输入的有效数据长度
密码文本框输入	<input type="password"…>	同上
多行文本框输入	<textarea name="" …> input text </textarea>	name 属性，设置输入控件的名字 rows 属性和 cols 属性，分别用来设置文本框的列数和行数，列与行以字符数为单位
button 按钮	<input type="button"…>	name 属性，设置输入控件的名字 value 属性，value 为按钮的显示名称
radio 单选按钮	<input type="radio"…>文本	name 属性，单选按钮名称，一般是若干个 radio 一组，取相同的 name，构成一组 value 属性，存储单选按钮的取值，多个具有相同 name 的单选按钮应该具有不同的 value checked 属性，设置该单选按钮默认时是否被选中
复选框	<input type="checkbox"…>	复选框是对某种输入做出"是"或"否"的选择。属性设置同 radio
复选列表框	<select name="" …> <option value="">…</option> <option value="">…</option> … </select>	name 属性，设置输入控件的名字 size 属性，设置下拉式列表的高度，默认值为 1，显示弹出式的列表框。若设置 size 的值大于 1，则不会有 PopUp 效果。如果 size 小于可选的项目数量，则出现垂直滚动条 <option>标记用来指定列表框中的一个选项，value 属性，用来给<option>指定的那一个选项赋值，这个值将传送到服务器，服务器通过调用<select>标记的 name 的 value 属性来获得该区域选中的数据项。selected 属性，用来指定默认选项

续表

标记/输入类型	一般形式	常用属性
隐藏元素	＜input type="hidden" …＞	在网页上并不显示，不需要用户输入，主要用于客户端和服务端的数据传送 name 属性，设置输入控件的名字 value 属性，存储文本框的取值
文件上传标记	＜input type="file" …＞	name 属性，设置输入控件的名字 size 属性，设置文本框长度 accept 属性，设置上载文件过滤，即单击“浏览”按钮时，只显示指定类型的文件列表
表单提交	＜input type="submit"…＞	将表单内容提交给服务器 value 属性，设置提交按钮的显示名字，一般为“确定”、“提交”等易于理解的名字 onclick 属性，设置表单提交前的处理函数、一般表单输入有效性验证
重填按钮	＜input type="reset"…＞	表单清除就是要将表单中已做的输入和选择全部清除，重新填写

对于上述表单及输入标记，说明如下。

(1) 每个输入元素都具有 name 属性和 value 属性，这是因为浏览器在进行 HTML 解析时，创建一个 HTML DOM 树，每个标记对应一个 DOM 树中的节点对象，标记的一般属性即为对象属性，事件属性则为对象的方法。标记的 name 属性是对象的名称，脚本程序可以通过对象名称访问和操作节点对象。value 属性存储节点对象的值。

(2) form 标记的 method 属性，决定客户端和服务端的 HTTP 通信中表单数据的传输方式，也决定服务端获取表单输入数据的方法。通常有两种方法。

① get 方法，将数据打包放置在环境变量 QUERY_STRING 中作为 URL 整体的一部分传递给服务器。QUERY_STRING 变量可存储量是有限的，一般限制在 1KB 以下。

② post 方法，通过 HTTP 的实体头域传递数据到 Web 服务器，没有数量限制。

(3) 如果表单中包含 file 类型输入，为了实现通过 http 上传文件，在表单标记＜form＞中需要加入编码方案属性 enctype="multipart/form-data"。该编码方案在传送大量数据时比默认的表单编码方案“application/x-url-encoded”效率更高。因为，URL 编码只有很有限的字符集，当使用任何超出字符集的字符时，必须用“%nn”代替(nn 表示两个十进制数)，因此，通过 URL 编码方式上载的文件大小将是原来的 2～3 倍。

例如，有如下代码：

```
< form name = "myForm" method = "POST" action = "/custom/feedback.jsp" enctype = "multipart/
form-data">
  提交论文：< input type = "file" name = "F1" size = "20">
</form >
```

显示结果为：

提交论文：[　　　　　　] 浏览...

用户单击“浏览”按钮，选择要提交的文件，文件将被上传到 Web 服务器。因为安全的原因，在 HTML 中，不能设置上传文件在服务器上的存储路径。上传文件的存储路径是在表单处理程序中设置的，在 Web 服务器端，通过组件，来设置每一个<input type="file">上传文件的存储路径。一个 HTML 表单可以设置多个<input type="file">控件，从而一次上传多个文件到 Web 服务器。

【例 3-7】 设计一个个人信息登记表页面，要求输入个人姓名、性别、出生日期，教育程度、工作单位、单位地址、电话、邮箱、个人简介、个人照片等信息。

分析：个人信息登记页面应由表单来实现，同时需要使用表格进行页面布局。设计的页面如图 3-4 所示。

图 3-4 个人信息登记表页面设计

代码清单：myinfo. htm

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<form name="form1" method="POST" action="myinfo.jsp" enctype="multipart/form-data">
<table border="1" width="600" id="table1">
<tr height="35">
  <td width="160">姓名: <input type="text" name="myname" size="14"></td>
  <td width="150">性别: <input type="radio" value="sex" name="male" checked>男<input
type="radio" name="sex" value="female">女
  </td>
  <td width="170">出生日期: <input type="text" name="birthday" value="2010-01-01"
size="10">
  </td>
  <td>教育程度:
    <select size="1" name="degree">
      <option value="college">本科</option>
      <option value="master">硕士</option>
      <option value="doctor">博士</option>
      <option value="other">其他</option>
    </select>
</td>
</tr>
```

```
<tr height="35">
    <td colspan="2">电话:<input type="text" name="tel" size="35"></td>
    <td colspan="2">邮箱:<input type="text" name="email" size="37"></td>
</tr>
<tr height="35">
    <td colspan="4">兴趣爱好:<input type="checkbox" name="interesting" value="sports">运
动<input type="checkbox" name="interesting" value="music">音乐<input type="checkbox"
name="interesting" value="other">其他
    </td>
</tr>
<tr height="120">
    <td colspan="4">个人简介<br><textarea rows="6" name="brief" cols="92"></textarea>
</td>
</tr>
<tr height="30">
    <td colspan="4">上传个人照片<input type="file" name="F1" size="65"></td>
</tr>
<tr height="35">
    <td colspan="4" align="center"><input type="submit" value="提交" name="B1">
  <input type="reset" value="重置" name="B2">
    </td>
</tr>
</table>
</form>
</body>
</html>
```

通过上面的例子,可以看出以下几点:①不是所有的 form 表单都需要提交到 Web 服务器处理,即可以没有 action 属性;②如果是用 FrontPage 的 table 进行页面布局,在单元格中插入 form 表单控件时,生成的 html 代码比较混乱,甚至会出现多个 form,此时,要手工调整 html 代码,只要将所有的输入控件包含在<form>…</form>标记对之间就可以了;③在<form>标记的前后会产生空行,可将<form>标记放在<table>标记的外面。

3.2.9 层叠样式表技术

在 HTML 中,大多数标记都包含了默认的显示样式,定义了所标记内容在浏览器中的布局和显示外观。如果要修改一个标记的默认显示属性,需要通过标记的 style 属性为标记的相应属性赋值,这使得标记更加复杂,难于维护。W3C 为了解决 HTML 的结构化问题和实现 Web 中的总体外观控制,于 1996 年底,公布了层叠样式表(Cascading Style Sheet,CSS)表单规范。所谓层叠是指对于容器元素指定的所有选项,将被自动地应用到其包含的所有元素中。

在 CSS 规范中,定义了非常精细的样式控制,不仅可以在文档的头部通过<style></style>标记来修改默认显示样式和定义样式,还可以在标记内使用 style 内联样式,或者定义样式类,通过 class 属性应用,这就在很大程度上实现了内容和显示的分离,便于页面的维护。

1. CSS 规范预定义样式属性

在 CSS 规范中，定义了大量的样式属性，利用这些属性可以对标记进行精细的设计，达到传统标记属性无法实现的功能和效果。

CSS 对网页的控制是通过 CSS 属性实现的，这些属性可以分为以下几类。

1）字体属性

字体属性用来控制文本的字体、字号等，如表 3-6 所示。

表 3-6　字体属性

属　性　名	功　　能	取　　值
font-family	字体	＜字体名＞
font-size	字体大小	＜absolute-size＞、＜relative-size＞、＜length＞、＜percentage＞等
color	文字颜色	＜颜色值＞
font-style	是否斜体	normal、italic、oblique（与 italic 相同）
font-weight	字体的粗细	normal、bold、bolder、lighter，也可以用数值：400 相当于 normal，700 相当于 bold
font-variant	小写字母改为用大写的小体字	normal、small-caps

说明： 字体大小的单位为长度单位，常用的单位有点数（pt）、像素（px）、pica（pc）、ex（小写字母 x 的高度）或 em（字体高度）、毫米（mm）、厘米（cm）、英寸（in）。在取值一栏中，写在"＜"和"＞"内的表示是一个用户输入的数，例如＜percentage＞表示可以输入一个百分数，例如 150%。其他的取值为常量。

2）颜色和背景属性

颜色和背景属性用来控制页面或元素的颜色、背景颜色、背景图片等，如表 3-7 所示。

表 3-7　背景属性

属　性　名	功　　能	取　　值
background-color	背景颜色	颜色值
background-image	背景图片	background-image：url（路径/文件名）
background-repeat	背景图片重复方式	repeat-x（沿水平方向平铺）、repeat-y（沿垂直方向平铺）、no-repeat（不重复）
background-attachment	背景滚动或固定	fixed（背景固定）、scroll（背景与页面一起滚动）
background-position	背景图片位置（仅背景图片不重复时有效）	＜percentage＞、＜length＞、top、left、right、bottom、center

3）文本间距与文本对齐属性

文本间距与文本对齐属性用来控制文本的间距，段落的间距、行高、缩进等，如表 3-8 所示。

表 3-8　文本间距与文本对齐属性

属　性　名	功　　能	取　　值
word-spacing	单词间距	normal、＜length＞
letter-spacing	字符间距	normal、＜length＞
text-decoration	文字的装饰样式	none、underline（下划线）、overline（上划线）、line-through(删除线)、blink(闪烁)
vertical-align	文字的垂直位置	basline(在标准位置)、sub(下标位置)、super(上标位置)、top(与对象顶端对齐)、text-top、middle、bottom(与对象底端对齐)、text-bottom、＜percentage＞(向上为正,向下为负,100%为垂直上移一行,-100%为垂直下移一行)
text-transform	文本转换为其他形式	capitalize(首字母大写)、uppercase（转换为大写)、lowercase(转换为小写)、none(无转换)
text-align	对齐方式	left、right、center、justify(对齐每行的文字)
text-indent	首行缩进	＜length＞、＜percentage＞
line-height	定义行高	normal、＜number＞、＜length＞、＜percentage＞

4）容器属性

容器是指其中可以包含内部元素、对象或数据的元素,如表格、单元格等,页面也是容器对象。CSS 中对这样的对象都统一用容器属性来控制。它包括以下子类:边距属性、填充属性、边框属性和图文混排属性。常用的属性如表 3-9 所示。

表 3-9　容器属性

属　性　名	功　　能	取　　值
边距属性(对象到周围对象的距离)		
margin	全部边距	auto、＜length＞、＜percentage＞
margin-top	顶端边距	auto、＜length＞、＜percentage＞
margin-bottom	底端边距	auto、＜length＞、＜percentage＞
margin-left	左侧边距	auto、＜length＞、＜percentage＞
margin-right	右侧边距	auto、＜length＞、＜percentage＞
边框属性		
border-style	边框样式	none(无边框)、dotted(打点边框,在大多数浏览器中会变为线)、dash(断裂边框,在大多数浏览器下也会变为线条)、solid(实线边框)、double(双线条边框)、groove(3D沟槽边框)、ridge(3D脊状边框)、inset(3D嵌入边框)、outset(3D突出边框)
border-top	顶端边框的三属性	border-top-width、border-style、border-color(同时给定三个属性值,属性值之间用空格分隔,下同)
border-bottom	底端边框的三属性	border-bottom-width、border-style、border-color
border-left	左侧边框的三属性	border-left-width、border-style、border-color
borer-right	右侧边框的三属性	border-right-width、border-style、border-color
border-width	一次定义边框宽度	thin、medium、thick、＜length＞
border-top-width	顶端边框宽度	thin(细)、medium(中)、thick(粗)、＜length＞

续表

属 性 名	功 能	取 值
border-bottom-width	底端边框宽度	thin、medium、thick、＜length＞
border-left-width	左侧边框宽度	thin、medium、thick、＜length＞
border-right-width	右侧边框宽度	thin、medium、thick、＜length＞
border-color	边框颜色	颜色值
衬距属性(内容到边框的距离)		
padding	全部衬距	＜length＞、＜percentage＞
padding-top	顶端衬距	＜length＞、＜percentage＞
padding-bottom	底端衬距	＜length＞、＜percentage＞
padding-left	左侧衬距	＜length＞、＜percentage＞
padding-right	右侧衬距	＜length＞、＜percentage＞
图文混排属性		
width	宽度属性	auto、＜length＞、＜percentage＞
height	高度属性	auto、＜length＞
float	文本环绕	left、right、none
clear	单边文本环绕	left、right、both、none

对于边框，有三个要素决定，即线宽、线型和颜色，例如 boder:1px soloid #ff0000，定义了边框为 1 个像素宽度、实线、红色的边框。此外，对于边距属性，margin-left、margin-right 可以取负数，例如＜span style＝"margin-left:-10px"＞＜/span＞，表示在正常输出位置左移 10 个像素，这可能会遮盖左边已有的内容，起一种特别效果。

5) 列表属性

列表属性是指用来对列表进行控制的 CSS 属性，如表 3-10 所示。

表 3-10 列表属性

属 性 名	功 能	取 值
display	定义是否显示	none(不显示)、block(以块的形式显示)、inline(以行内元素的形式显示，元素的前后不带有换行)、list-item(以列表的形式显示)
white-space	列表项中空格的处理方式	normal(忽略空格和换行符)、pre(保留空格和换行符)、nowrap(列表项不自动换行，直到出现＜br＞ 标记)
list-style-type	在列表前加项目符号	disc(实心圆)、circle(空心圆)、squre(实心方块)、decimal(阿拉伯数字)、lower-roman(小写罗马字母)、upper-roman(大写罗马字母)、lower-alpha(小写英文字母)、upper-alpha(大写英文字母)、none(无)
list-style-image	在列表项前加图案	格式：url(imageURL)、none
list-style-position	列表项中第二行的起始位置	inside(列表项目符号在列表项文本以内，列表项文本与项目符号对齐)、outside(默认值，列表项目符号放在文本以外，列表项文本自身对齐)
list-style	一次定义前面的列表属性	list-style-type 或 list-style-image、list-style-position (同时给定两个属性值，属性值之间用空格分隔)

6）鼠标状态属性

鼠标状态属性用来控制鼠标指针的形状，如表 3-11 所示。

表 3-11　鼠标状态属性

属　性　名	功　　能	取　　值
cursor	指针效果	auto(自动)、crosshair(定位“十”字)、default(默认指针)、hand(手形)、move(移动)、e-resize(箭头朝右方)、ne-resize(箭头朝右上方)、nw-resize(箭头朝左上方)、n-resize(箭头朝上方)、se-resize(箭头朝右下方)、sw-resize(箭头朝左下方)、s-resize(箭头朝下方)、w-resize(箭头朝左方)、text(文本“I”形)、wait(等待)、help(帮助)

7）定位属性

在 CSS 中，提供了一组定位属性，可以对块级元素和行内元素进行定位，改变正常的输出流结果，精确控制页面上的对象的位置，如表 3-12 所示。

表 3-12　定位属性

属　性　名	功　　能	取　　值
position	定义位置	absolute(绝对)、relative(相对)、static(静态)
left、top	水平位置和垂直位置	<length>、<percentage>、auto
width、height	对象的大小、宽和高	<length>、<percentage>、auto
overflow-x overflow-y	内容超出时的处理	auto、visible、hidden(不显示滚动条)、scroll
clip	剪切	auto、shape（形状，有效的设置为 rest（top，right，bottom，left））
z-index	设置叠放次序	auto、<integer>，当 position 定位有层叠时，z-index 值大的在上面，默认为 0，可取负数
visibility	设置可见性	visible(可见)、hidden(隐藏)、inherit(继承外层元素的显示属性)

在 HTML 的输出中，<div>、<p>或<h1>元素被称为块级元素，这些元素显示为一块内容，即“块框”。与之相反，<span>等元素称为“行内元素”(元素前后没有换行)。它们的内容显示在行中，即“行内框”。CSS 为定位和浮动提供了 position、left、right、top、bottom 等属性，利用这些属性，对元素在页面的输出中进行定位，定位的基本思想就是允许定义元素框相对于其正常位置应该出现的位置，或者相对于父元素、另一个元素甚至浏览器窗口本身的位置。从而建立列式布局，将布局的一部分与另一部分重叠，完成通常需要使用多个表格才能完成的任务。

关于 position 属性，说明如下。

(1) relative，元素框相对于原先正常输出偏移某个距离。元素仍保持其未定位前的形状，它原本所占的空间仍保留。

(2) absolute，元素框从文档流完全删除，并相对于其包含块进行定位。包含块可能是文档中的另一个元素或者是初始包含块。元素原先在正常文档流中所占的空间会关闭，就好像元素原来不存在一样。元素定位后生成一个块级框，而不论原来它在正常流中生成何

种类型的框。

(3) fixed,元素框的表现类似于 absolute,只是其包含块是视窗本身,即浏览器窗口。

2. 样式表与 CSS 选择符

在 HTML 中,除<script>等个别标记外,几乎所有的文档体内的标记都有其默认的显示样式,要修改标记的默认显示样式,有多种实现方法:①在页面的头部(<head></head>)内部,使用<style>…</style>标记,定义样式表,这样定义的样式只在该网页发生作用,称为嵌入式样式;②在标记中通过 syles 属性进行定义,称为内联样式表;③将样式表定义保存为单独的样式表文件,然后在网页中引用该文件,在标记内使用通过 class 属性或 id 属性进行设置。

样式表定义的一般形式是:

```
<style type="text/css">
  选择符{属性名:属性值; 属性名: 属性值;… }
</style>
```

所谓"选择符",可以简单地理解为对一个样式表的命名,不同的选择符决定了样式的使用范围和使用方式。在 CSS 中,选择符分为六类。

1) HTML 选择符

选择符为标记名,此时样式定义将替换标记的默认显示,任何 HTML 标记都可以作为 CSS 选择符。

```
<style type="text/css">
  body {color: #ff0000; background: black; }
  p{margin:2px 4px}
</style>
```

上述样式表定义,将修改<body>和<p>标记的默认显示样式。

2) 类选择符

HTML 选择符虽然修改了标记的默认显示样式,但是同一标记的显示样式是一样的。如果希望同一标记有不同的样式,可以将样式表定义成一个个的样式类,选择符就是样式类的名字,称为类选择符。样式类可以关联一个标准的标记,也可以用于任何标记。定义样式类的语法格式为:

```
[HTML 标记].<类选择符> {
    属性: 属性值
    [;属性: 属性值 … ]
}
```

例如:

```
.word1 {color: lime; background: #ff80c0 }
p.warning { font-weight: bolder; color: red; background: white }
```

其中,word1 类不隶属于任何标记,可以用于任何 body 元素,因为它在样式表中没有和具体的 HTML 元素关联。但 warning 类只能用于段落标记<p>。

当定义了样式类后,类选择符在标记中通过 class 属性引用。例如:

```
<p class = warning>警告: </p>
<p class = word1 > Please turn off the power first</p>
```

3) ID 选择符

如果一个样式表在同一个页面中只出现一次,可以定义为 ID 选择符,其定义方法与类选择符相似,只是在选择符的前面用"#"而不是"."。在 HTML 规范中,id 属性是一个标记的标识,通过 id 属性可以访问一个标记或对其进行操作。一个页面中,不能有两个相同的 id。在 HTML 规范中,如果从大的设计出发,我们会发现有些内容是许多页面都有的,例如页面的标题、页面的脚注等。对于这些公有的部分,可以直接设置它们的内容和显示方式,这就是说通过它的 id 来设置样式。

和使用标记的 class 属性相比,在布局思路上,一般坚持这样的原则:id 是先确定页面的结构和内容,然后再为它定义样式;而 class 相反,它先定义好一类样式,然后在页面中根据需要把类样式应用到不同的元素和内容上面。在实际应用时,class 更多地被应用到文字版块以及页面修饰等方面,而 id 更多地被用来实现宏伟布局和设计包含块或包含框的样式。

例如:

```
#location - table{
  border:0px;
  cellspacing:0px;
  cellpadding:0px;
}
#help - table{
  border:1px solid #0163A2;
  cellspacing:5px;
  cellpadding:px;
  font - family:宋体;
  font - size: 12px;
  color:#FF0000;
  line - height:150%;
}
#foot - table{
    margin - top:15px;
    height:65px;
    font - family:宋体;
    font - size:12px;
    color:#FFFFFF;
    text - align:center;
    line - height:150%;
    background - color:#d0d0cf;
}
```

上述 id 设定用于每一个页面中的当前位置、帮助信息以及版权信息的设定。在每一个页面中,这是一样的。

当将样式表定义成 id 选择符后,在标记中通过 id 属性设置为 id 选择符即可,需要注意

的是，在同一个网页中，标记的 id 值应该是唯一的，不能重名。

4）关联选择符

关联选择符是由两个或更多的单个选择符组成的串，由于层叠顺序的关系，关联选择符的优先权更大。关联选择符只对选择符的最内层元素起作用，对单独的外层元素无定义。定义关联选择符时，单独的选择符之间用空格分开。

例如：

```
table a{color:red}
```

它定义了<table>内的超链接标记<a>的样式，对于<table>标记本身，以及<table>标记外的<a>没有影响。

5）组合选择符

将多个选择符共用一个样式表定义，避免样式表定义的重复，这就是组合选择符。定义组合选择符，单个选择符之间用西文的“,”分割。

例如：

```
h1,h2{color:red}
```

则<h1>和<h2>标记定义为红色。

6）伪元素选择符

它主要定义超链接标记<a>，目的是定义不同状态下同一个 HTML 元素的形态。例如，单击前、单击后、光标移动到超链接文本上时，这些不同的状态超链接显示不同。

例如，要个性化超链接的显示，可以定义下面的样式。

```
<style type="text/css">
    a {color:#0000FF;font-size:12px;text-decoration:none}
    a:link{
        color:#00FF00;
        text-decoration:none
    }
    a:visited{
        color:#00FF00;
        text-decoration:none
    }
    a:hover {
      color:#FF0000;
      font-weight:bold;
      text-decoration:none
    }
    a:active {
      color:#0000FF;
      text-decoration:none
    }
</style>
```

这样，对于文档中的超链接（<a>…），文字显示为蓝色，当鼠标指针指向的时候则显示红色，同时字体也加粗显示，不显示下划线。对于访问过的页面，超链接显示为绿色，

指针指向时,颜色不变。

特别说明:在 CSS 定义中,a:hover 必须位于 a:link 和 a:visited 之后,才能生效;a:active 必须位于 a:hover 之后,才能生效。

要修改一个标记的默认属性,需要知道这个标记有哪些属性,记住一个标记的所有属性是很困难的,这需要借助于 FrontPage、Dreamweaver 等网页制作工具来定义样式。例如,在 FrontPage 中,通过"格式"→"样式"命令,可以修改标记的默认样式,或者新建样式类,创建 css 文件等。

3. 标记的 style 属性与内联样式

要修改标记的默认显示样式,还可以设置标记的 style 属性来实现,这种修改只针对当前一个标记,不影响其他同名的标记,从而实现标记按照特定的形式显示,这就是内联样式。

例如:

```
<p style = "color: red; font - family:'Impact'">红色英文 Imapct 字,如果字体可用的话.</p>
```

要使用内联样式,必须在文档的<head>…</head>部分包括以下标记:

```
<meta http - equiv = "Content - Type" content = "text/css">
```

内联样式可以改变标记的默认显示样式,如果需要多个标记使用同样的样式显示时,需要为每一个标记添加 style 属性,非常麻烦。另外,内联样式和需要展示的内容混合在一起,显得比较混乱,修改不够方便,这本身也是 HTML 规范的一大弊端。

总之,style 属性可以改变标记的默认显示样式,采用 style 属性管理大量文档的显示将十分困难。当管理人员需要改变某些标记的显示形式时,需要做大量的微调工作。要解决站点级的显示问题,内联样式将非常困难,必须借助于样式表。

4. 样式表(.css)文件

在 HTML 文档的<head>…</head>标记内,通过<style>…</style>定义的样式、id 样式还是 class 样式类,只能应用于当前的 HTML 文档。如果要将这些样式应用到其他 HTML 文档中,应该使用样式文件。即,将这些样式定义存储在一个扩展名为. css 的样式文件中,css 文件可以是一个标准的 HTML 文件,只不过<body>…</body>为空。然后,当某个网页需要使用其中的样式时,在文档的<head>…</head>中增加<link>标记,一般形式如下:

```
<link type = "text/css" rel = "stylesheet" href = "mystyle.css">
```

这样,在当前文档中,就可以使用样式文件 mystyle. css 中定义的样式了。

一种良好的 HTML 页面就是充分利用 CSS 技术,将页面的显示和布局分开,从而保证页面维护的灵活性。

【例 3-8】 设计表格样式,绘制线宽为 1 的表格线,并设计表格的标题行单元格样式。

分析:在 HTML 中,表格是由一个个的单元格组成的,每个<table>标记和<td>标记都有边框,因此,即使设置表格的 cellspacing="0",表格的边框也是两个像素的线宽。要实现一个像素的表格线,需要设计<table>和<td>的边框 css 属性。

（1）样式表文件 mytable.css。

代码清单：mytable.css

```
.table-hasframe
{
    margin-top:15px;
    border-left:1px solid #0163A2;
    border-bottom:1px solid #0163A2;
    font-size:12px;
    background-color:#FFFFFF
}
.cell-title
{
    border-top:1px solid #0163A2;
    border-right:1px solid #0163A2;
    font-weight:bold;
    text-indent:0px;
    line-height:150%;
    text-align:center;
    vertical-align:middle;
}
.cell-normal
{
    border-top:1px solid #0163A2;
    border-right:1px solid #0163A2;
    padding-left:5px;
    text-indent:0px;
    vertical-align:middle;
}
```

在上述的css文件中，定义了一个表格样式，定义了其上边距、左边框和下边框。两种单元格样式类，都定义了上边框和右边框。这样只要在应用上述样式类时，设置<table>的cellspacing="0"，即可实现1个像素的表格线。

（2）在页面中应用样式表文件。

代码清单：mytable.htm

```
<html>
<head>
<meta http-equiv="Content-Language" content="zh-cn">
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<link rel="stylesheet" type="text/css" href="mytable.css">
</head>
<body>
<table class="table-hasframe" width="100%" cellspacing="0">
    <tr height="35">
        <td class="cell-title" width="70">序号</td>
        <td class="cell-title" width="100">留言人</td>
        <td class="cell-title">标题</td>
    </tr>
    <tr height="30">
```

```
            <td class = "cell - normal" align = "center"> 1 </td>
            <td class = "cell - normal" align = "center">袁蓉蓉</td>
            <td class = "cell - normal">在 CSS 中,利用 id 和 class 设置标记样式有什么不一样?
</td>
        </tr>
        <tr height = "30">
            <td class = "cell - normal" align = "center"> 2 </td>
            <td class = "cell - normal" align = "center">郭晶</td>
            <td class = "cell - normal">如何设置段落的行距?</td>
        </tr>
</table>
</body>
</html>
```

在浏览器中打开上述页面,显示结果如图 3-5 所示。

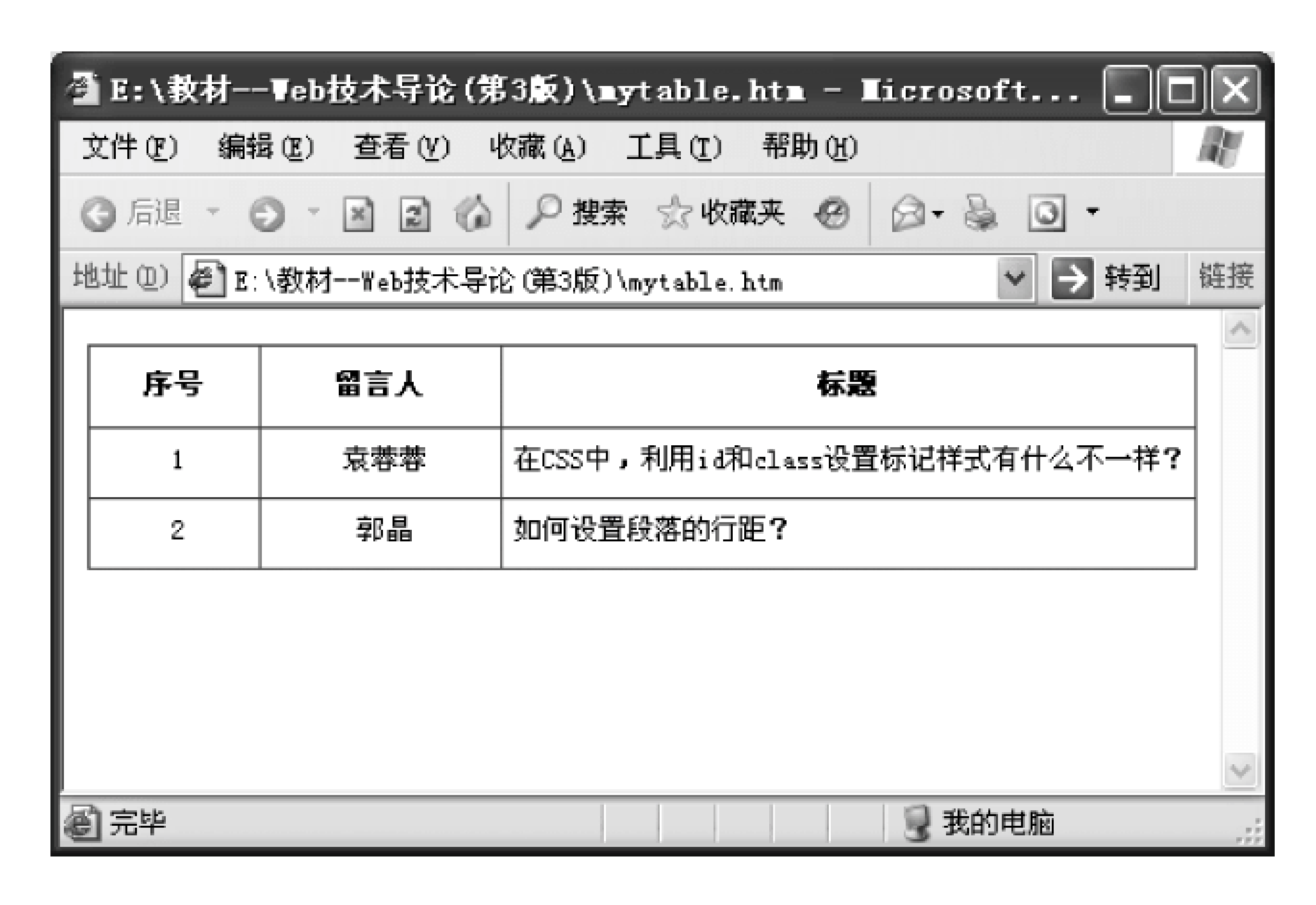


图 3-5　一个像素的表格线表格

3.2.10　<div>标记和<span>标记

为了更好地实现元素的精确定位,浏览器厂商在 HTML 规范基础上,在网页中增加了层和位置的概念。通过层次块的显示、隐藏和移动来实现灵活的页面显示效果。同时,为了动态地改变页面上一些区域的显示,还引入了<span>标记,它是一种类似行内占位的标记。和其他 html 元素不同,<div>和<span>没有默认的显示样式,因此,用户必须通过 style 属性或 class 属性来设置样式,来构造丰富的页面布局。

1. 图层标记<div>

层次块标记<div>…</div>用于定义网页上的一个矩形块,中间可以包含引起行中断的标记,如<table>标记等。层次块标记的一般形式是:

```
<div id = "idname" style = "divstyle" >
</div>
```

主要属性有:

(1) id 属性，用于标识一个<div>块，通过 id 可以引用该块或对图层进行操作。

(2) style 属性，定义图层块的位置、大小、显示属性等。

此外，图层还可以接受 onclick 等鼠标事件，来增加交互功能。

通常情况下，在网页上，可以用<div>和</div>标记来定义一个矩形区域，即定义一个图层块，通过图层块的 style 属性操作，来得到一些特殊的效果。通过客户端程序，可以实现对区域的显示、隐藏和移动等操作。如果没有层次块，要实现一个区域，例如一个 table 的移动是很不方便的。

【例 3-9】 图层的显示或隐藏。

分析：在许多 Web 应用系统的页面设计中，通常使用图层来显示一些帮助信息，通过单击来显示或隐藏图层。例如，我们在开发 GSL 系统时，设计的页面如图 3-6 所示。



图 3-6　图层的显示和隐藏设计

当用户在“教学计划列表”文本上单击时，显示“系统帮助”图层，再次单击，隐藏该图层。示例代码如下：

```
<html>
<head>
<meta http-equiv = "Content - Type" content = "text/html; charset = gb2312">
<link rel = "stylesheet" type = "text/css" href = "pubcss/linestable.css">
<link rel = "stylesheet" type = "text/css" href = "pubcss/common.css">
<script language = "javascript">
function showorhide_div(id)
{
   if (document.all.item(id).style.display == "none")
     document.all.item(id).style.display  =  "block";
   else
     document.all.item(id).style.display  =  "none";
}
</script>
</head>
<body>
<table class = "location - table" width = "97 % ">
<tr height = "35">
    <td class = "location - title" colspan = "2">您的位置: 教学计划管理><a href = "#"
onclick = "showorhide_div('helpdiv')">教学计划列表</a></td>
</tr>
<tr height = "35">
    <td align = "right">
```

```
        <a href = "addbook - groupslist.jsp">添加班级</a> |
        <a href = "addbook - personadd.jsp">添加单个学生</a>
      </td>
</tr>
</table>
<div id = "helpdiv" style = "border:0px solid #0163A2;padding:5px;display:none">
<table class = "banner - table" style = "width:97%;margin - bottom:20px" cellpadding = "0"
border = "0">
<tr height = "20">
   <td class = "menuposword" width = "97%" align = "left">系统帮助:</td>
   <td rowspan = "2">
         <a href = "#" onclick = "showorhide_div('helpdiv')"><font face = 'Wingdings' color =
'#FF0000' style = 'font - size:32pt'>r</font></a>
   <td>
</tr>
<tr class = "bannerword" height = "20">
   <td align = "left">一项教学计划对应一个教学班,教学班可以包括多个不同的物理班;对于选
修课,一个教学计划的教学班,可以是某些班的部分同学.每一个教学计划都有一个唯一的教学计划
号,教学计划号 = 课程代码 + 课程顺序号.
   </td>
</tr>
</table>
</div>
<table border = "1" width = "97%" id = "table1">
<tr>
     <td> </td>
     <td> </td>
</tr>
</table>
</body>
</html>
```

上述代码中,涉及脚本程序和CSS技术,这在接下来的小节中介绍。在默认状态下,图层占整个窗口的宽度,且不显示图层区域的边界,可以设置style="border:1px solid #0163A2"来显示图层的边框,以便对图层有个更直观的认识。

2. 区段标记<span>

<span>元素不同于一般的HTML元素,它没有自己的显示内容和显示特性。引入<span>标记的目的是定义一个行内区域,可以给这个区域一个id属性,结合CSS,对该区域设定特定的样式,以得到需要的显示效果。也可以使span标记的区域响应鼠标或键盘事件,制作菜单、树形结构目录、列表框及下拉列表框项目。和<div>标记相比,<span>在CSS定义中属于一个行内元素(元素前后没有换行),而<div>是块级元素,我们可以通俗地理解为<div>为大容器,<span>就是小容器,在一个<div>中可以包含若干span元素。

【例3-10】 <span>标记应用示例。

分析:在Web应用的开发中,有时候需要动态地改变页面上一些地方的内容,比如单元格、文本内部等。对于单元格,它是由<td>标记定义的,因此可以访问,但如果是一段文

本的一部分，则没有特别的标记就无法访问了，此时，可使用<span>标记标记需要操作的文本。

代码清单：exa3-10 . htm

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<script language="javascript">
function setspan(spid,flag)
{
   if(flag==1)
   {
     //使用函数参数访问 span 对象
     document.all.item(spid).innerHTML="<font color=red size=7>Hao</font>";
     //直接使用<span>标记的 id 访问对象
     sp2.style.display="inline";
     sp3.style.display="none";
   }
   if(flag==2)
   {
     document.all.item(spid).innerHTML="<font color=red size=7>Jiang</font>";
     sp2.style.display="none";
     sp3.style.display="inline";
   }
}
</script>
</head>
<body>
<a href=javascript:setspan('sp1',1)>Mr.Hao</a></br>
<a href=javascript:setspan('sp1',2)>Miss.Jiang</a></br>
<p>亲爱的<span id=sp1>xxx</span><span id=sp2 style="display:none">先生</span><span
id=sp3 style="display:none">女士</span></p>
</body>
</html>
```

运行上述代码，显示结果如图 3-7 所示。



图 3-7 <span>标记的应用

3.2.11 脚本程序标记和对象标记

在 HTML 网页中，除了一些基本的文本、图片等内容外，还可以插入脚本程序、Java 程序、内嵌对象、ActiveX 控件等，从而制作出更加新颖、丰富和交互能力更强的 Web 页面。这些新的文档内容需要通过<object>、<script>、<applet>、<bssound>、<embed>等

标记来进行标记,下面介绍其中的脚本程序定义标记和插入对象标记。

1. 脚本程序定义标记<script>…</script>

用户可以在 HTML 文件中插入脚本程序,脚本程序分为客户端脚本程序和服务端脚本程序两类。客户端脚本程序一般为 JavaScript 程序,在客户端浏览器中解释执行,可以在文件头和文档体内书写脚本程序。服务端脚本程序与 Web 服务器有关,服务端脚本通常是在文档体内定义。不同的 Web 服务器应该对应不同的服务端脚本程序,因为服务端脚本程序是在 Web 服务器上运行的。

在 HTML 文件中标记脚本程序的一般形式是:

```
<script language = "" runat =  "" >
    脚本程序代码
</script>
```

属性 language 用于设置脚本程序语言,runat 属性设置脚本是客户端还是服务端脚本,默认为客户端脚本。要声明是服务端脚本,则属性值设为 runat="Server"。服务端脚本程序也可以使用其他的标记,如 Tomcat 支持的服务器脚本标记为<%…%>标记。

2. 插入对象标记<object>…</object>

插入对象包括 Falsh 动画、ActiveX 组件或其他对象。在 HTML 中插入一个对象,需要使用<object>…</object>标记来标记对象,在<object>标记内,还需要使用多个<param>标记,来设置该对象属性的初始值,即为该对象传递参数值。一般形式如下:

<object>标记的一般形式是:

```
<object classid  = "" id = "obj1">
   <param >
   <param >
     . . .
</object >
```

通常情况下,<object>标记以及每一个子标记<param>都包含了许多属性,插入对象的类型不同,这些参数的设置悬殊较大。对于各种类型对象的详细介绍略。

3.2.12 帧与浮动帧

帧(Frame)即网页中的一个区域,用来将浏览器窗口划分为多个区域(子窗口),每个子窗口中装载一个 HTML 文件。即每个 HTML 文件占据一个帧,而多个帧可以同时显示在同一个浏览器窗口中,这样的 Web 页面称为框架网页。

在 HTML 中,有两种帧技术,一种是帧(frame),另一种是浮动帧(iframe)。

1. 帧

帧页定义的一般形式是:

```
<frameset rows = "" cols = "">
```

```
  <frame name = "frame - name" target = "target - window" src = "src - uri">
  <frame name = "frame - name" target = "target - window" src = "src - uri">
  …
</frameset>
```

其中<frameset>…</frameset>标记定义帧，必须有一个 rows 属性或 cols 属性，将屏幕分成若干行或若干列。然后跟着是相应的每一帧定义，由<frame>来标记，<frame>为单标记。如果某个<frame>进一步进行了拆分，在<frame>处，可以嵌套<frameset>…</frameset>。

1）<frameset>…</frameset>标记

<frameset>…</frameset>标记对放在帧的主文档的<body></body>标记对的外边，也可以嵌在其他帧文档中，并且可以嵌套使用。此标记对用来定义主文档中有几个帧并且各个帧是如何排列的。它具有 rows 和 cols 属性，使用<frameset>标记时这两个属性至少必须选择一个，否则浏览器只显示第一个定义的帧。

rows 用来规定主文档中各个帧的行定位，而 cols 用来规定主文档中各个帧的列定位。这两个属性的取值可以是百分数、绝对像素值或星号（“ * ”），其中星号代表那些未被说明的空间，如果同一个属性中出现多个星号则将剩下的未被说明的空间平均分配。同时，所有的帧按照 rows 和 cols 的值从左到右、从上到下排列。

2）<frame>标记

<frame>标记放在<frameset>…</frameset>之间，用来定义一个具体的帧。<frame>标记具有 src 和 name 属性，这两个属性都是必须赋值的。src 是此帧的源 HTML 文件名（包括网络路径，即相对路径或网址），浏览器将会在此帧中显示 src 指定的 HTML 文件；name 是此帧的名字，这个名字用来供超文本链接标志<a href="" target="">中的 target 属性指定链接的 HTML 文件将显示在哪一个帧中。

例如，定义了一个帧，名字是 main，在帧中要显示的 HTML 文件为 mint. htm，则代码是<frame name="main" src="mint. htm">。如果有一个链接，在单击了该链接后，需要在 main 的帧中显示文件 newone. htm，则代码为“链接的文本”。这样一来，就可以在一个帧中建立网站的目录，加入一系列链接，当单击链接以后在另一个帧中显示被链接的 HTML 文件。

此外，<frame>标记还有 scrolling 和 noresize 属性，scrolling 用来指定是否显示滚动条，取值可以是 yes（显示）、no（不显示）或 auto（若需要则会自动显示，不需要则不显示）。noresize 属性直接加入标记中，不需赋值，它用来禁止用户调整一个帧的大小。

【例 3-11】 设计一个框架页面，要求框架宽度为 1024，在宽屏时始终居中显示。

分析：目前由于显示器的分辨率一般都大于等于 1024，因此，设计页面时，基本上以 1024 作为页面宽度，如果用户的屏幕是宽屏的，页面应居中显示，而不是拉宽或靠左。

代码清单：myframe. htm

```
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http - equiv = "Content - Type" content = "text/html; charset = gb2312" />
</head>
<frameset cols = " * ,1024, * " framespacing = "0" frameborder = "0" id = "mainframes">
```

```
    <frame name="free-left" scrolling="no" noresize src="#">
    <frameset rows="102,*" framespacing="0" frameborder="0" id="frames01">
      <frame name="topframe" scrolling="no" noresize src="mybanner.html">
      <frameset cols="200,10,*" id="frames02">
        <frame name="leftframe" scrolling="auto" noresize src="menu.htm">
        <frame name="ctrlleft"  scrolling="no" noresize src="scrollleft.html">
        <frame name="mainFrame" scrolling="auto" noresize src="introduction.htm">
      </frameset>
    </frameset>
    <frame name="free-right" scrolling="no" noresize src="#">
  </frameset>
  </html>
```

在 myframe.htm 中，首先将屏幕分成了三列“*,1024,*”，中间为 1024，这样就保证了页面始终显示为 1024 宽度。在中间的 1024 区域，又分成了两行“102,*”，上面为页面的 banner，下面剩余高度为页面主体内容。对于主体内容区，又分成了三列“*200,10,*”。在 FrontPage 中，设计视图如图 3-8 所示。



图 3-8　框架网页在浏览器中的显示

上述框架结构设计可以应用于一般的 Web 应用系统的功能首页设计，左侧为菜单，中间为折叠控制，右侧为工作区。

下面是 scrollleft.html 页面代码，控制左侧菜单帧的折叠。

代码清单： scrollleft.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<base target="_self">
<style>
```

```
body{margin:0 auto; font-size:12px; text-align:center; font-family:宋体, Helvetica, sans-serif; background:#C0C0C0;}
</style>
<script language="JavaScript">
var tt = 1;
function ctrl_left()
{
   if (tt == 1)
   {
     window.parent.frames02.cols = "0,10, * ";
     lrarrow.src = "toctrlleft01.gif";
     tt = 0;
   }
   else
   {
     window.parent.frames02.cols = "200,10, * ";
     lrarrow.src = "toctrlleft02.gif";
     tt = 1;
   }
}
</script>
</head>
<body style="margin-left:0px;cursor:hand" onClick="ctrl_left()">
<table height="100%" width="20" border="0" cellspacing="0" bordercolor="#CCAA00" bgcolor="#c0ddeb">
<tr>
   <td width="10" align="center">
       <img id="lrarrow" src="../toctrlleft02.gif">
   </td>
</tr>
</table>
</body>
</html>
```

2. 浮动帧

浮动帧是用<iframe>标记的帧，通常利用表格布局，在一个单元格中插入浮动帧。一般形式是：

```
<iframe id="frame-id" name="frame-name" src="page-uri" style="height:100%; width:100%;visibility: inherit; z-index:2;"></iframe>
```

使用浮动帧同样可以制作出类似上面帧的效果，这可以通过显示或隐藏所在单元格实现折叠。

3.3 扩展标记语言基础

在互联网的发展历史上，有两种非常核心的技术——Java 和 XML。Java 提供了程序代码的平台无关性，而 XML 则保证了数据的平台无关性，已成为 Web 应用中数据表示和数据交换的标准。但是，人们对 XML 的认识远远没有对 HTML 的认识彻底和清晰，有些

理解甚至是完全错误的。那么,究竟什么是XML呢?XML和HTML有什么不同,它们的本质区别是什么呢?同时,由于XML毋庸置疑的优越性,XML的不断发展壮大,挂在“XML”一词下的标准和规范不断变化,了解这些标准的来龙去脉,以及它们之间的关系,对于掌握XML也是至关重要的。

3.3.1 XML技术简介

XML和HTML都称为标记语言,但是两者有着本质的不同。HTML是一种数据展示技术,它对内容加以标记,使得内容以特定的方式在Web浏览器中显示。但是,XML的基本动机则是对数据结构的表达,实现内容和内容展示的分离,其追求的目标是XML文档的结构良好以及内容的有效性约束。而XML文档内容的显示,则需要其他相应的XML规范,例如XML扩展样式语言(XSL)等。

1. HTML的不足

HTML的出现无疑是Internet技术和Web技术的一次突破,为推动Internet和Web技术的发展发挥了巨大的作用。随着Web技术的快速发展,Internet上的Web信息越来越多,内容越来越复杂,数据格式也越来越多,HTML已经无法满足表达日益丰富的数据形式的需要。主要表现在以下几个方面。

(1) HTML的标记固定,HTML只是一种表现技术,不能表达语义。

(2) 不能适应现在越来越多的网络设备和应用的需要,比如手机、PDA、信息家电都不能直接显示HTML。

(3) HTML代码不规范、臃肿,浏览器需要足够智能和庞大才能够正确显示HTML。

(4) HTML文档中数据与表现混杂,页面要改变显示,就必须重新制作HTML。

2. XML的产生和发展

HTML的不足推动了XML的产生和发展。1996年8月,那些关心SGML的专家聚集在美国西雅图,成立了一个名为GCA(Graphic Communications Association,图形通信协会)的组织,研究如何开发SGML以便它适应和促进Web技术的发展。他们对SGML难于被理解和实现的方面进行简化,去掉其语法定义部分,适当简化DTD部分,并增加了部分互联网的特殊成分。为了体现它与HTML的不同,工作组将其命名为XML,同时也将自身更名为XML工作组。1998年2月10日,XML工作组正式向W3C提交了XML的最终推荐标准,这就是XML 1.0标准。

在XML中,SGML的最初动机得以延续,那就是将文件内容和处理这些内容的应用程序进行分离,在文件内容中不嵌入数据的处理过程代码,文件内容被编码为条理清晰的文本,从而便于数据交换和处理。对数据进行研究有着重要的意义,因为,数据往往是相对稳定的,变化的通常是处理这些数据的程序。实现数据和操作这些数据的程序的分离是XML的设计动机,这是深刻理解XML的基础。在XML中,如果XML某方面设计得与应用程序太过紧密,就可以认为这是一种Bug,这是使用XML最重要的一个原则。

XML标准的发展没有HTML那样迅速,直到2002年10月15日,W3C才发布了XML 1.1候选推荐标准。在XML 1.0规范中,使用的字符集为Unicode 2.0。随着

Unicode 版本的升级，XML 1.1 支持新的 Unicode 字符，不再局限于一个具体的 Unicode 版本。此外，在 XML 1.1 中，增加了 IBM 大型主机规定的换行符(#x85：十六进制的 85)和 Unicode 换行符(#x2028)的处理能力，这些改变都提高了 XML 的国际化支持水平。

3. XML 解析器

XML 文档需要在 XML 解析器中才能运行，XML 解析器是一个软件模块，应用程序利用它来解析 XML 文档并且得以访问数据和数据结构。XML 解析器有确认型和非确认型两种。确认型 XML 文档解析器检查 XML 文档的语法，将 XML 文档内容同文档类型定义(DTD)和 Schema 作比较。XML 分析器通过判断 XML 数据是否和预定义的确认规则相符，判定 XML 文档是否构造良好。非确认型 XML 解析器也进行 XML 文档语法的检查，但不进行 XML 文档内容和 DTD 及 Schema 的比较。

现在几乎所有的主流浏览器都内置了 XML 解析器，支持 XML 和 XSLT。例如，在微软的 Internet Explorer 浏览器中，内置了 XML 确认性解析器 MSXML。如果是一个有效的 XML 文件，在浏览器中打开 XML 文件时，XML 文档将显示为一个树状结构，称为 XML 文档树。通过单击元素左侧的加号或减号，可以展开或折叠元素结构。此外，浏览器或其他应用程序也可以使用 css 或 xslt 样式转换显示 XML 文档数据。

4. XML 技术分析

随着 Internet 的快速发展，尤其是电子商务、Web 服务等应用的广泛使用，XML 类型的数据成为当前主流的数据形式。虽然，XML 作为一种通用的数据交换语言，已经成为业界的一种具有垄断性的标准，在跨平台、跨系统数据交换方面拥有无可比拟的优势。但是，和关系数据库相比，XML 的最大缺陷就是它的效率较低。因为在关系型数据库中，数据的字段名只需要出现一次，但是在 XML 数据文件中，元素名将反复出现，这必然会影响到查询的效率。

作为一种数据存储方案，XML 技术无疑具有绝对的优势。提高数据的查询、使用效率成为 XML 技术研究的热点，这些研究包括 XML 与关系数据库之间的相互转换，利用关系数据库的成熟技术对 XML 数据进行处理。为提高 XML 的查询效率，需要为 XML 类型提供索引功能。如果对 XML 文档不构建索引结构，那么针对 XML 数据的任何查询都很可能导致对整个文档树的遍历，随着 XML 数据集的增大，这种开销是不可忍受的。

上述关于 XML 研究和应用的需求推动了 XML 的发展，也导致了许多新的 XML 相关标准和规范的产生，包括 XML Schema、扩展样式语言 XSLT、XML 路径语言(XPath)、XML 查询语言(XQuery)、XML 链接语言规范 XLink 和 XPointer 等。这些规范都以 XML 语法为基础，遵守 XML 规范，因此，目前 XML 已经成为各种语言规范的元语言。

3.3.2 XML 文档结构

XML 文档是一个纯文本文件，XML 规范定义了 XML 文档良好的结构，将一个 XML 文档分为 XML 文档头部(文档序言)和文档体(文档内容)两个部分，一般形式如下：

```
<?xml version = "1.0" encoding = "GB 2312"?>
```

```
<?xml-stylesheet type="text/xsl" href="userxslfile.xsl"?>
<!DOCTYPE rootElementName SYSTEM "userDTDfile.dtd">
<!DOCTYPE rootElementName[
    <!ELEMENT elementname(element-defination)>
    …
]>

<rootElementName>
   <elementname>ElementContent</elementname>
   <elementname>ElementContent</elementname>
    …
   <elementname>ElementContent</elementname>
</rootElementName>
```

1. XML 文档头部

XML 文档头部又称为文档序言,它是指根元素之前的部分,XML 文档头部的作用是通知 XML 解析器按相关条件和限制对 XML 文档进行解析。在 XML 文档中,文档头部包括 XML 声明、处理指令、文档类型定义和注释四部分。按 XML 规范要求,"声明"必不可少,并且作为文档的第一条语句出现,其他部分根据需要确定,可以省略。

1) XML 声明

XML 声明是 XML 文档头部的第一条语句,也是整个文档的第一条语句。XML 声明语句的格式如下:

```
<?xml version="versionNumber"  [encoding="encodingValue"][standalone="yes/no"] ?>
```

XML 声明语句,以"<? xml"开始,以"? >"结束,表示这是一个 XML 文档。Version 属性是必选属性,表明使用的 XML 规范的版本号,以便解析器进行正确的解析。encoding 属性指定本 XML 文档使用的字符集。XML 解析器至少能够识别 UTF-8 和 UTF-16 两种编码,英文用 UTF-8 编码。如果使用简体汉字,应设置 encoding 属性值为 GB 2312。若指定繁体汉字赋值为 BIG5。standalone 属性指定本 XML 文档是否需要外部的 DTD 文档作为本文档的校验依据,即本 XML 文档是否为一个独立文档。默认值是 yes,表示是独立文档不需要外部 DTD 关联,否则应该赋值为 no。

2) 处理指令

处理指令(Process Instruction,PI)是在 XML 文档中由应用程序进行处理的部分,XML 解析器把信息传送给应用程序,应用程序解释指令,按照它提供的信息进行处理。处理指令是以"<?"开始,以"? >"结束,其格式是:

```
<?处理指令名称 处理指令信息?>
```

以"xml-[name]"开头的处理指令指定的是[name]中给出的与 XML 相关的技术。处理指令以"xml-stylesheet"开头,指明相关的技术是样式表,这条指令就被传送给由 type 指定的引擎,而不传给 XML 解析器。例如,<? xml-stylesheet type="text/css" href="file1.css" ? >,该语句中 type 表示关联的文档类型,现在的文档是文本、层叠样式表类型引擎,href 指定关联的文档所在位置和文档的名称。再如,<? xml-stylesheet type=

"text/xsl" href="file1.xsl" ? >，则关联的样式文件为扩展样式表文件。以上是使用较频繁的两条指令，它们规定 XML 文档中的数据使用哪一种格式在浏览器中显示。

3）文档类型定义

如果 XML 文档需要使用 DTD 对内容进行有效性验证，则需要在文档头部增加外部文档类型定义声明或进行内部文档类型定义。外部文档类型定义声明，即声明一个外部 DTD 文件，它属于处理指令的范畴。一般形式是：

```
<!DOCTYPE rootElementName SYSTEM "dtdFile ">
```

在外部文档类型定义后面将是内部文档类型定义(DTD)部分，用于定义 XML 中用到的元素、元素属性和实体，即声明用户自定义标记及相关属性，其目的是用于确认型 XML 解析器检查 XML 文档是否结构良好。相对于用外部的 DTD 声明，这里的文档类型定义称为内部 DTD 声明。其一般形式是：

```
<!DOCTYPE rootElementName [
  <!ELEMENT element - name(element - defination)>
   …
]>
```

其中，rootElementName 为文档的根元素，在根元素内定义其他元素。例如，<! ELEMENT address(buildingnumber, street, city, state, zip)>，则定义一个名称为 address 的元素，address 元素按顺序又包含<buildingnumber>、<street>、<city>、<state>、<zip> 五个元素。可以进一步定义<buildingnumber>元素为<!ELEMENT buildingnumber(#PCDATA)>，它确定了<buildingnumber>元素的取值。

在一个 XML 文档中，内部文档类型定义(DTD)部分不是必需的，如不需要可以省略。

4）注释

XML 中使用注释对文档进行解释说明，增加程序的可读性，处理程序不对注释标记的内容进行处理。与 HTML 一样，注释是由“<!--”开始，由“-->”结束，注释语句的格式是：

```
<! -- 注释文字 -->
```

使用注释时需要注意，第一，注释可以出现在文档头部，也可以出现在主体部分，但不能出现在声明之前，声明语句必须是 XML 文档的第一条语句；第二，注释可以包容标记，使标记失去作用，但注释不能出现在标记中。

2. XML 文档内容

XML 文档内容，即 XML 文档体，它是 XML 文档的数据部分。文档体包括一个或多个元素，每个元素由一个开始标记和一个结束标记定义。文档体中的元素定义了数据的结构，一个单独的根元素包含所有其他元素，文档中所有数据都包含在文档体的根元素中。

在文档体内，还可以使用名称空间，即在每一个元素和属性前面加上前缀“名称空间：”来唯一地标识一个元素或一组元素的属性，以避免多个 XML 文档中的元素重名。

【例 3-12】 一个简单的 XML 文档。

下面是 Brion 给 Jane 的便条，使用 XML 格式，内容如图 3-9 所示。

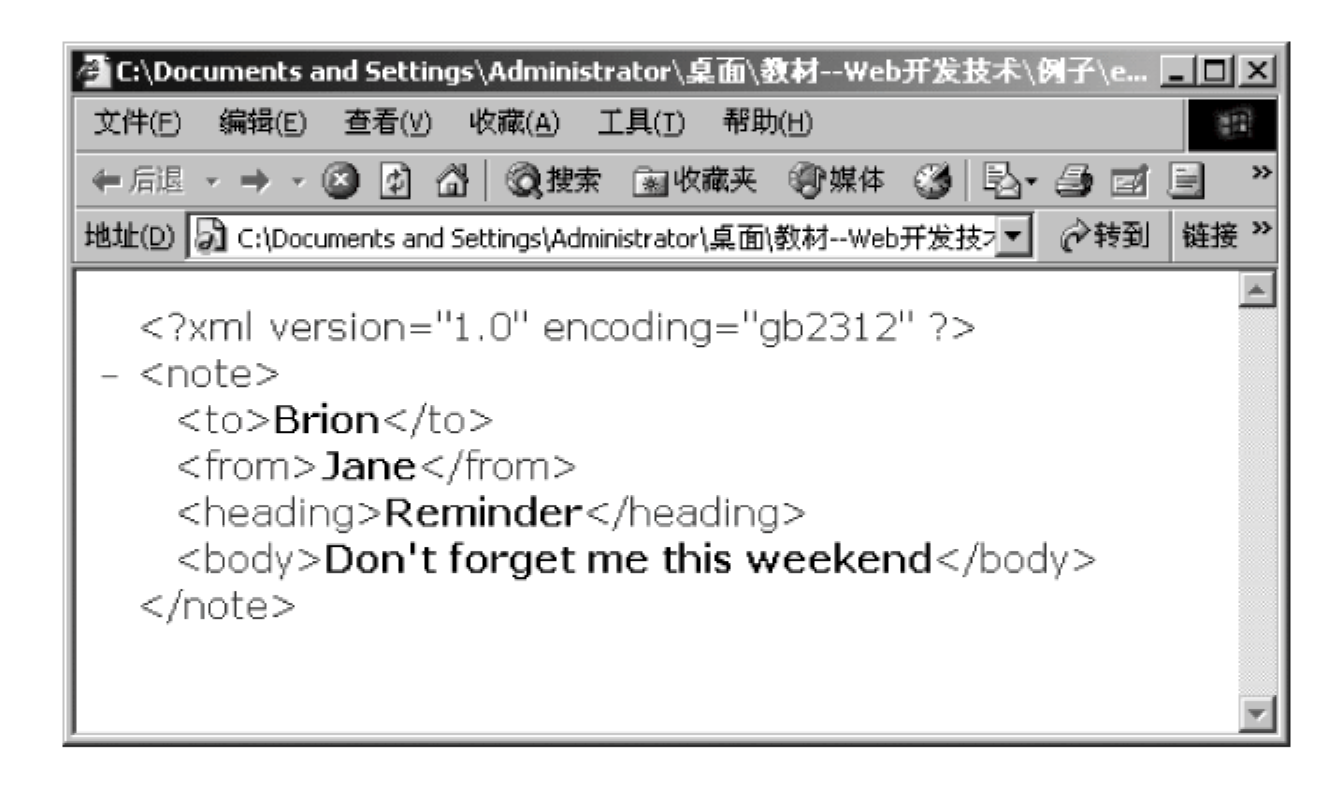
用 html 的思想，双击该文档在浏览器中打开，显示如图 3-10 所示。

在浏览器中，我们看到了一棵 XML 文档树。这个 XML 文档做了什么呢？好像什么都没做。我们需要的是观念上的转变，不能再以 html 的思想来理解 xml 了。XML 不是 HTML 的替代品，HTML 设计的目的是用来显示数据，重点是显示数据以及如何使数据的显示更美观。XML 是用来存储数据的，XML 设计的目的是用来描述数据结构，以及存储数据，实现数据存储和显示的分离，数据的显示则是通过层叠样式表或样式转换语言实现的。



```
<?xml version="1.0" encoding="gb2312" ?>
<note>
  <to>Brion</to>
  <from>Jane</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend</body>
</note>
```

图 3-9　一个简单的 XML 文档



```
<?xml version="1.0" encoding="gb2312" ?>
- <note>
    <to>Brion</to>
    <from>Jane</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend</body>
  </note>
```

图 3-10　XML 文档在浏览器中的显示界面

3.4　文档类型定义

在 XML 中，没有像 HTML 一样拥有一个通用的标记集合，标记(在 XML 中，称为“元素”)是通过文档类型定义(Document Type Difinition，DTD)来实现的。DTD 定义了 XML 文档中可以使用的元素符号、元素的属性、元素的排列方式/顺序、元素能够包含的内容等，其目的是保证确认型 XML 解析器来确定 XML 文档数据的有效性，保证 XML 文档结构良好。DTD 可以在 XML 文件中直接定义，也可以保存在一个完全独立的文件(.dtd)中。因此，DTD 分为内部 DTD(在 XML 文件中直接定义 DTD)和外部 DTD(在 XML 文件中调用已经编辑好的 DTD 文件)两种。

3.4.1 在 DTD 中声明 XML 元素

一个内部 DTD 声明必须写在 XML 文档的头部，在处理指令和 XML 根元素之间。一般形式为：

```
<!DOCTYPE rootElementName[
    <!ELEMENT element-name (element-definition) >
    …
]>
```

其中，<! DOCTYPE 表示开始设定 DTD。rootElementName 指定此 DTD 的根元素的名称，一个 XML 文件只能有一个根元素。

<! ELEMENT element-name (element-definition) > 为元素定义语句，其中，<! ELEMENT 是 XML 的保留字，表示开始元素定义。element-name 是为元素所起的名称，element-definition 是对元素的定义，就是说<元素>…</元素>之间能够包含什么内容。元素的内容可以是一般性文字，也可以是其他元素。

元素内容定义(element-definition)可以是：

```
EMPTY | #PCDATA | 元素 | ANY
```

说明如下。

(1) EMPTY，没有内容的元素。在 XML 文件中，空元素不需要结束标记，但必须采用</空元素名>这样的写法。XML 中的空元素类似于 HTML 中的单标记，空元素不标记内容，但可以设置元素属性。如果元素定义为 EMPTY，EMPTY 不需要用小括号括起来，即元素定义语句可写为：

```
<!ELEMENT element - name EMPTY>
```

(2) #PCDATA，声明一个基本元素，元素内容为可解析字符数据(parsed character data)，即元素所标记的内容将被 XML 解析器解析。如果内容可能是子标记、实体等，它们一并被解析器解析，从而得到正确的数据关系。

解析器之所以这么做是因为 XML 元素可包含其他元素。例如，定义一个<name>元素如下：<! ELEMENT name(#PCDATA)>，则对于下列的标记内容：

```
<name><first>Bill</first><last>Gates</last></name>
```

解析器会把<name>元素的内容解析为下列的关系：

```
<name>
 <first>Bill</first>
 <last>Gates</last>
</name>
```

如果不希望解析器解析，则使用 CDATA，意思是字符数据(character data)，CDATA 是不会被解析器解析的文本，在这些文本中的标签不会被当做标记来对待，其中的实体也不会被展开。

(3) 元素，声明一个容器元素，即元素还可以包含另外的元素，形成一种嵌套和层次结

构。声明容器元素的基本语法为：

```
<!ELEMENT containerElement(containedElement_1,…, containedElement_n)>
```

其中，containerElement 为容器元素名称，$containedElement_1$ 至 $containedElement_n$ 为被包含的元素。被包含元素可以取下列三种格式之一。

① element，要求该元素有且只有一个值。

② element+，要求该元素有一个或多个值。

③ element*，要求该元素有零个或多个值。

例如，＜！ ELEMENT 书籍（名称，作者，价格）＞，表示定义了一个容器元素(即标记)“书籍”，包含三个子元素，分别是“名称”、“作者”和“价格”。

在一些容器元素的声明中，有可能它包含的子元素是多个子元素中的一个，那么在声明此父元素时，就可以把它声明成选择性元素，可供选择的子元素用“|”分隔。例如：＜! ELEMENT 配偶（妻子|丈夫）＞。

(4) ANY，表明所有可能的元素以及可解析的数据。

【例 3-13】 文档类型定义 DTD 举例。

对于例 3-12 中的文档内容进行 DTD 定义，XML 文档(note. xml)内容为：

```
<?xml version = "1.0" encoding = "gb2312" ?>
<!DOCTYPE note[
<!ELEMENT note (to,from,heading,body)>
<!ELEMENT to       (#PCDATA)>
<!ELEMENT from     (#PCDATA)>
<!ELEMENT heading  (#PCDATA)>
<!ELEMENT body     (#PCDATA)>
]>
<note>
    <to>Brion</to>
    <from>Jane</from>
    <heading>Reminder</heading>
    <body>Please give me a call tonight</body>
</note>
```

3.4.2 在 DTD 中声明元素属性

和 HTML 标记一样，XML 元素往往也包含属性。在 XML 中，定义元素属性的一般形式是：

```
<!ATTLIST element-name attribute-name Type Default-value>
```

其中，＜! ATTLIST 表示开始属性的设定，element-name 为要定义属性的元素名，attribute-name 是元素属性名称，Type 是该属性属性值的类别，元素属性值类型见表 3-13。

Default-value 是指该属性值的取值特点，有四种不同的属性取值，分别是：

(1) #REQUIRED，表示在标记中必须给定属性值。

(2) #IMPLIED，表示该属性值可以省略。

表 3-13 XML 中元素属性类型

类 型 名	描 述
CDATA	表明属性值可以是任何文本字符串，但不包括小于号(<)和西文双引号(")，如要使用这两个符号可以使用实体引用"<"和"""
ENUMERATED	枚举该属性的取值范围，一次只能有一个属性值能够赋予属性
NMTOKEN	表示属性值只能由字母、数字、下划线、.、:、-等符号组成
NMTOKENS	表示属性值能够由多个 nmtoken 组成，每个 nmtoken 之间用空格隔开
ID	该属性在 XML 文件中是唯一的
IDREF	表示该属性值是参考了另一个元素的 id 属性
IDREFS	表示该属性值是参考了多个其他元素的 id 属性，这些 id 属性的值用空格隔开
ENTITY	表示该属性的设定值是一个外部的 entity，如一个图片文件
ENTITIES	该属性值包含多个外部 entity，不同的 entity 之间用空格隔开
NOTATION	属性值是在 dtd 中声明过的 notation(声明用什么应用软件解读某些二进制文件，如图片文件)

(3) #FIXED，表示一个固定的属性值。

(4) 字符串，指定属性的默认取值。

下面是一组 XML 元素和元素属性声明：

```
<!ELEMENT FAMILY (PERSON+)>
<!ELEMENT PERSON EMPTY>
<!ATTLIST PERSON
        myID     ID #REQUIRED
        name     CDATA #REQUIRED
        sex      (male|female) "male"
        nickname NMTOKENS #IMPLIED
        parentID IDREFS #IMPLIED
>
```

上述 XML 语句为元素 PERSON 声明了 myID、name、sex、nickname 和 parentID 五个属性。myID 属性类别为 ID，表明 myID 属性的取值在此 XML 文件中是唯一的，否则将出现解析错误。此属性设定中的属性取值要求为 #REQUIRED，表示 myID 属性在元素 PERSON 中必须出现，否则也会产生解析错误。

属性 name 为 CDATA 属性类别，表明属性取值为一般性文字。属性 sex 的属性值类别是枚举类型，取值范围为 male 或者 female，如果在 XML 文件中没有为此属性赋值，属性默认取值是一个字符串 male。

nickname 属性类型为 NMTOKENS，规定了其取值的字符集，此属性可以省略。属性 parentID 的类型为 IDREFS，表明该属性的值必须在文档中出现过，该属性可以省略。如果该属性的值没在文档中出现过，解析器将认为该文档为不规范文档。

根据上面的元素属性说明，我们看下面的 XML 文档数据：

```
<FAMILY>
<PERSON myID="P_1" name="Brion" nickname="sun@#$"/>
<PERSON myID="P_2" name="Jane" sex="female"/>
<PERSON myID="P_3" name="Linda" sex="female"/>
```

```
<PERSON myID = "P_4" parentID = "P_1 P_5" name = "David"/>
</FAMILY>
```

上述文档数据是不正确的,因为在第一个元素 PERSON 的 nickname 属性值中包含了 NMTOKENS 所不允许的字符"@ # $"。此外,parentID 属性值中出现了值“P_5”,但该值没有在文档中出现过。

需要说明的是,许多 Web 浏览器(例如 IE),仅支持 XML 文档的结构良好,并没有数据内容的 DTD 有效性验证功能,因此,在 XML 文档的语法出错时,浏览器打开文档时会报错,但内容如果不符合 DTD 定义,浏览器不报错。

对文档的结构和内容的有效性检查,可通过 Altova XMLSpy 工具来完成,打开 XML 文档,在“XML”菜单中,包含“检查良构性”和“验证 XML”两条菜单命令,第一条用于检查结构是否良好,第二条则检查内容是否有效。例如,对于上述文档,按 F8 键,在信息输出窗口显示如下检查结果:

```
⊟ ⊗ 文件E:\教材--Web技术导论(第3版)\family.xml无效。
  ⊟ ! NMTOKENS属性'nickname' DEF 的值'sun@#$'里的'sun@#$'必须遵循名称符号的句法规则。
     错误位置: FAMILY / PERSON / @nickname
```

3.4.3 定义实体

在 XML 的 DTD 中,还可以定义实体(Entity)。实体实际上起一种类似“宏”的作用,一些常用的或者不便于直接书写的文字或数据,可以用一个标识定义下来,在数据中可以直接引用,这就是实体。实体的引用通过“&”来引用,末尾加“;”。

在 XML 中,有 5 种预定义实体,分别是字符“&”(&)、“＜”(<)、“＞”(>)、“""”(") 和“'”(')。除了这些预定义实体,还允许用户自己定义实体,例如,如果在 XML 文档中需要频繁使用词组 Good Luck,可以在 DTD 中这样表示: ＜! ENTITY gl "Good Luck"＞。这样当使用 Good Luck 时,可以输入 ≷,从而可以避免拼错和重复输入相同的信息,这里,gl 就是实体。

在 XML 中,实体可以分成内部实体、外部实体和参数实体三种类型。内部实体的一般形式为:

```
<!ENTITY   entityName   "will be replaced string">
```

如果被替换的文本很长,可能要把被替换的信息存储在一个文件中。可以通过外部实体参考来实现,即在实体名和文件的 URL 中使用关键字 SYSTEM,构成外部实体。一般形式为:

```
<!ENTITY   entityName   SYSTEM   "URL">
```

例如,下面是一个关于实体定义和引用的 XML 文档的代码:

```
<?xml version = "1.0" encoding = "gb2312" ?>
<!DOCTYPE message[
<!ELEMENT title (#PCDATA)>
<!ENTITY hi "您好!">
<!ENTITY ans "&hi; 谢谢!你也好吗?">
```

```
]>
<message>
<title>Jane,&hi;,&ans;</title>
</message>
```

除此之外，在 XML 中，还提供了参数实体，它在实体定义中通过在实体名前插入百分号(%)实现，百分号表示该实体为参数实体。一旦被定义，参数定义可以通过用百分号和分号包围参数名来实现。例如：<! ENTITY % role"(boss | manager | employee)">。

最后，我们要说明的是，对于定义面向数据的文法，DTD 的功能，特别是有效性验证，远不如 XML Schema，但是 DTD 的实体是 XML Schema 很难做的。在 DTD 中很容易定义实体，但是这种功能很难在 XML Schema 中再现。一般情况下，实体常见于叙述性文法，在这个领域 DTD 的地位仍然很稳固。

3.4.4 字符数据段

通过预定义 XML 实体可以在 XML 文档中加入特殊符号，如果需要大量的特殊符号，可以使用字符数据段，称为 CDATA 段。

字符数据段(CDATA 段)是指不被 XML 解析器解析的文本段，即使这些文本包含了元素标记、实体引用以及特殊符号，它们均不被解析器解析。CDATA 段可以使用户在一个 XML 文档中引用大量的特殊符号文本块，而不需要分别以实体的形式来代表每一个特殊字符。比如 JavaScript 代码，包含大量“<”或“&”字符，在被 XML 解析时，必须使用预定义实体，否则将发生错误。为了避免错误，减少麻烦，可以将脚本代码定义为 CDATA。

CDATA 部分中的所有内容都会被解析器忽略，CDATA 部分由 "<! [CDATA[" 开始，由 "]]>" 结束，一般形式为：

```
<![CDATA[
  text
]]>
```

其中，text 是包含特殊字符的文本串，该文本不被 XML 分析器检查。XML 处理器负责分析或者以一种有意义的方式使用该文本块。其中的左右方括号(“[”和“]”)不能省略。

例如，下面是一个包含 CDATA 的 XML 文档：

```
<?xml version="1.0" encoding="gb2312" ?>
<people>
<![CDATA[
<teacher>
<name>Li</name>
<sex>女</sex>
<age>25</age>
<add>Shan Da Nan lu 27 号</add>
</techer>
]]>
</people>
```

在浏览器中打开该文档时，CDATA 段内的内容不被 XML 解析器解析为 XML 元素。

3.4.5 声明并保存外部 DTD 文件

如果希望将 DTD 声明应用到其他 XML 文件中，应该将 DTD 文本保存为一个独立的 .dtd 文件，这样一个 DTD 就可以被多个文档引用，保证版本的一致。对于 .dtd 文件，除了没有内部 DTD 中的<! DOCTYPE rootElement[…]>语句外，其他和一个 XML 文件相同。而且有关元素数目、排列顺序、空元素设定、选择性元素、属性设定、Entity 声明等都和内部 DTD 相同。

当创建 DTD 文件后，在一个 XML 文档中，可以引用外部 DTD 中声明的元素。要应用一个外部 DTD 文件，需要在 XML 文档的序言中，在 XML 文档内部 DTD 定义和文档内容之前，添加下列外部 DTD 文件声明语句：

```
<!DOCTYPE rootElementName SYSTEM "dtdFile">
```

其中，rootElementName 为应用 DTD 文件的 XML 文档根元素，dtdFile 为要引用的外部 DTD 文件名(.dtd)。

【例 3-14】 文档类型定义 DTD 举例。

下面是描述一篇学术论文的 XML 文档，定义了一个 paper 元素及相关属性。

```
<?xml version="1.0" encoding="gb2312" ?>
<!DOCTYPE paper[
<!ELEMENT paper (title,author+,abstract?,contents*)>
<!ATTLIST paper
   paperID CDATA #REQUIRED
   paperStatus (reviewing|accepted) "accepted"
>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT abstract (#PCDATA)>
<!ELEMENT contents (#PCDATA)>
<!ENTITY KP "knowledge points">
]>
<paper paperID="p177">
<title>The Research on a Kind of Knowledge Network for Self-learning</title>
<author>XW Hao</author>
<abstract>The self-learning is completely student-centered...</abstract>
<contents>The idea of self-learning is based on knowledge network mainly relies on manual
construction of &KP; ...
</contents>
</paper>
```

上述代码产生一个严格的 paper 结构，根元素 paper 声明为一个容器元素，包含一个 title 元素、至少一个 author 元素、一个可选的 abstract 元素、零个或多个 contents 元素。在声明 paper 元素后，紧接着声明 paper 元素的属性，为了便于阅读，属性的声明往往直接跟在元素声明的后面，虽然这不是必需的。接下来是声明一组基本元素，即 title、author、abstract 和 contents 元素。最后定义了一个实体 KP。

3.4.6 DTD的优势和不足

文档数据类型定义(DTD)在保证XML文档数据的有效性方面提供了一定的手段,定义DTD有如下两个方面的优点。

(1) 每一个XML文档都可携带一个DTD,用来对该文档格式进行描述,测试该文档是否有效的XML文档。通过定义公用外部DTD,多个XML文档可以共享该DTD,使得数据交换更为有效。甚至在某些文档中还可以使内部DTD和外部DTD相结合。在应用程序中也可以用某个DTD来检测接收到的数据是否符合某个标准。

(2) 对于XML文档而言,虽然DTD不是必需的,但它为文档的编制带来了方便,加强了文档标记内参数的一致性,使XML语法分析器能够确认文档。如果不使用DTD来对XML文档进行定义,那么XML语法分析器将无法对该文档进行确认。

虽然如此,DTD有其内在的不足,主要表现在:DTD有自己的特殊语法,其本身不是XML文档;DTD只提供了有限的数据类型,用户无法自定义类型;DTD不支持域名机制。这些不足导致了XML Schema的产生。但是,由于现在很多的XML应用是建立在DTD之上的,能读懂DTD文件以及在必要时创建简单的DTD文件仍然是很重要的。

3.5 XML Schema及其应用

在XML 1.0规范中,虽然DTD实现了XML文档中元素(标记)及其类型的定义,对于XML文档的结构化起到了很好的描述作用。但是,DTD支持的数据类型非常有限、扩展性较差,没有一种机制保证数据的应用方和数据的服务方对数据解释的一致。为此,W3C于2001年5月正式发布了XML Schema的推荐标准,提出了XML架构(Schema)的概念,通过XML Schema给XML数据标注数据类型,从而使得数据的应用方可以通过XML Schema规范对所交换的XML数据中的信息进行正确的解释,并实现自动处理。利用XML Schema规范,Web服务方和应用方无须事先协调所要交换的数据类型,从而解决了XML文档数据的结构和类型一致性解析问题,使XML文档更加结构良好。

3.5.1 XML Schema的概念

XML Schema是XML的应用,是一种对XML文档的内容和结构进行描述和定义的语言。在XML中,DTD和Schema都是XML文档数据的约束规则。和DTD相比,Schema具有以下优点。①XML Schema对DTD进行了扩充,引入了数据类型、命名空间,从而使其具备较强的可扩展性。DTD提供的数据类型只有CDATA、Enumerated、NMTOKEN、NMTOKENS等十种内置(built-in)数据类型。这样少的数据类型通常无法满足文档的可理解性和数据交换的需要。XML Schema则不同,它提供了更加丰富的数据类型,如long、int、short、double等常用的数据类型。②XML Schema利用Namespace将文档中特殊的节点与Schema说明相联系,一个XML文档可以有多个对应的Schema,而一个XML文档只能有一个对应的DTD。③XML Schema文档本身也是XML文档,而不是像DTD一样使用特殊格式。开发人员可以使用相同的工具来处理XML Schema和其他XML信息,而不

必专门为 Schema 使用特殊工具。

经过数年的大规模讨论和开发,如今 XML Schema 已经成为全球公认的 XML 环境下首选的数据建模工具。在应用中,有两种常用的 XML Schema,即 W3C 的 XSD Schema (XML Schema Definition)和微软的 XDR(XML-Data Reduced Language)。不同的 Schema 规范,虽然 Schema 的思想相同,但其名称空间所提供的预定义元素、内置数据类型不同。XDR 由微软最先提出,并且经过修订后在其软件中应用,所以 XDR 被广泛使用。目前支持 XDR Schema 的产品有 Microsoft Biztalk Server、Microsoft SQL Server 2000、Microsoft Office 2000、Microsoft IE 5.0 和后续版本。此外 XDR 也得到了 Extensibility 的 XML Authority 编辑工具的支持。但是,如果 XML 数据具有很强的开放性,如面向互联网应用,要考虑到对 XML 数据的约束规则今后可以被外部应用所兼容,应尽量选用 W3C 的 XSD Schema 规范。

3.5.2 XML Schema 文档结构

XML Schema 本身就是一个 XML 文件,不同的是,Schema 文件所描述的是引用它的 XML 文件中的元素和属性的具体类型,即对于 XML 文档中元素的定义。一个 XSD 文档就是一个命名空间,可以被其他的 XML 实例文档引用。

W3C XML Schema 是使用最为广泛的 XML Schema 架构,其 XML Schema 定义 (XML Schema Ddefination,XSD)的一般形式为:

```
<?xml version = "1.0" encoding = "gb2312"?>
<xs:schema xmlns:xs = http://www.w3.org/2001/XMLSchema
          elementFormDefault = "qualified"
          attributeFormDefault = "unqualified">
.
.(数据类型定义、元素声明、属性声明)
.
</xs:schema>
```

定义 Schema 的 XML 文档扩展名为.xsd,文档的根元素一定为<schema>,用于声明该 XML 文档是一个 XSD Schema 文档。元素<schema>包含若干属性,常用的属性如下。

(1) xmlns,规定在此 schema 定义中使用的一个或多个命名空间的 URI 引用,它通常是 W3C 规范的 XSD Schema。可以使用多个命名空间,对于每一个命名空间,通常分配一个前缀(即名称空间的别名),以便引用该命名空间中的各个 schema 组件而不至于重名。

如果某个命名空间没有命名,则写为 xmlns="命名空间",该空间为默认命名空间,使用它的元素无须前缀。

(2) targetNamespace,设置该 schema 的命名空间的 URI 引用。指出当 XML Schema 实例文档在引用该 schema 时,xmlns 设置的名称空间,用于在该 schema 的 XML 实例文档中使用该 Schema 中的数据类型和元素。

(3) elementFormDefault、attributeFormDefault,可选。指出任何 XML 实例文档在使用在此 schema 中声明过的元素或属性是否必须被命名空间限定。如果设置为 unqualified,则使用该 shchema 中的元素/属性时,无须限定前缀;如设置为 qualified,则在使用该 schema 中的元素或属性时,必须通过命名空间前缀限定。两属性的默认值均为

unqualified。

在利用 XML Spy 工具创建 W3C XML Schema 时，在<schema>元素中，自动添加一个元素。一般形式是：

```
<xs:element name = "ENTER_NAME_OF_ROOT_ELEMENT_HERE">
    <xs:annotation>
        <xs:documentation>Comment describing your root element</xs:documentation>
    </xs:annotation>
</xs:element>
```

该元素没有实际意义，只需要输入一个根元素的名字，它将在使用该架构的 XML 实例文档中作为 XML 文档的根来使用。如果一个架构应用于多个不同的 XML 实例文档，可以在架构中定义多个这样的根元素，供不同的 XML 文档使用。接下来用户可以定义简单数据类型定义、复杂数据类型定义、元素和属性声明、属性组定义等，这些定义或声明，就构成了一个用户的 XSD Schema，然后用户在自己的 XML 实例文档中就可以引用这个 schema 了，来确保 XML 文档的有效性。

3.5.3　XSD 内置元素与数据类型

在 W3C XML Schema 规范中，预定义了大量的 XML 元素和数据类型，供用户定义自己的 schema 时使用。这类似于程序设计语言中的关键字和标准函数。要设计自己的 schema，必须了解这些内置的元素和类型，否则将无法完成 schema 中数据类型和元素的定义。

1. XSD 预定义元素

在 W3C XML Schema 文档中，我们已经看到了<xs:schema>元素和<xs:elment>元素，这就是在命名空间 http://www.w3.org/2001/XML Schema 中定义的元素，称为预定义元素，这些预定义元素是我们设计自己 schema 的基础。

W3C XML Schema 规范定义了大量的元素和数据类型，常用的预定义元素见表 3-14。

表 3-14　W3C XML Schema(XSD)预定义元素

元素名	说 明	主要属性	内容	父元素
schema	定义 schema 的根元素	id、xmlns、elementFormDefault、attributeFormDefault、targetNamespace	include、import、redefine、notation、annotation、simpleType、complexType、element、attribute、group、attributeGroup	
import	向一个文档添加带有不同目标命名空间的多个 schema	id、namespace、schemaLocation	annotation	schema
include	向一个文档添加带有相同目标命名空间的多个 schema	id、schemaLocation	annotation	schema

续表

元素名	说 明	主要属性	内容	父元素
redefine	在当前 schema 中重新定义从外部架构文件中获取的简单和复杂类型、组和属性组	id、schemaLocation	simpleType、complexType、group、attributeGroup、annotation	schema
notation	描述 XML 文档中非 XML 数据格式	id、name、public、system	annotation	schema
annotation	标记 schema 注释	id	appinfo、documentation	任何元素
appInfo	规定 annotation 元素中应用程序要使用的信息	source	任何格式正确的 XML 内容	annotation
documentation	定义 schema 中的文本注释	source、xml:lang	任何格式正确的 XML 内容	annotation
simpleType	定义一个简单类型，规定与具有纯文本内容的元素或属性的值有关的信息以及对它们的约束	id、name	annotation、restriction、list、union	schema、element、attribute、 list、restriction、union
restriction	定义对 simpleType、simpleContent、complexContent 的约束	id、base	minExclusive、minInclusive、maxExclusive、maxInclusive、totalDigits、fractionDigits、length、minLength、maxLength、enumeration、whiteSpace、pattern	simpleType、simpleContent、complexContent
list	把简单类型定义为指定数据类型的值的一个列表	id、itemType	annotation、simpleType	simpleType
union	定义多个 simpleType 定义的集合	id、memberTypes	annotation、simpleType	simpleType
complexType	定义复杂类型	id、 Name、 abstract、mixed、block、final	annotation、simpleContent、complexContent、group、all、choice、sequence、attribute、attributeGroup、anyAttribute	schema、element、redefine
all	规定子元素能够以任意顺序出现，每个子元素可出现零次或一次	id、minOccurs、maxOccurs	annotation、element	complexType、group、restriction、simpleContent、complexContent、extension

续表

元素名	说 明	主要属性	内容	父元素
sequence	要求子元素必须按顺序出现，每个子元素可出现 0 到任意次数	id、minOccurs、maxOccurs	annotation、element、group、choice、sequence、any	complexType、choice、sequence、group、restriction、simpleContent、complexContent、extension
choice	仅允许在 <choice> 声明中包含一个元素出现在包含元素中	id、minOccurs、maxOccurs	annotation、element、group、choice、sequence、any	complexType、group、 choice、sequence、restriction、simpleContent、complexContent、extension
simpleContent	对 complexType 元素的扩展或限制且不包含任何元素	id、any attributes	annotation、restriction、extension	complexType
complexContent	定义对复杂类型（包含混合内容或仅包含元素）的扩展或限制	id、mixed、any attributes	restriction、simpleContent、extension、annotation	complexType
extension	扩展已有的 simpleType 或 complexType 元素	id、base、xml:lang	annotation、group、all、choice、sequence、attribute、attributeGroup、anyAttribute	simpleContent、complexContent
any	在复杂类型设计中 <any> 意味着在 XML 实例文档中，可以被任意数量的元素扩展	id、minOccurs、maxOccurs、namespace processContents	annotation	complexType、choice、sequence
anyAttribute	在复杂类型中扩展属性，即在元素中可以使用没有声明的属性。但这个属性应该用<attribute>声明	Id、namespace、processContents	annotation	complexType、restriction、simpleContent、complexContent、attributeGroup、extension
element	定义元素	id、 name、 type、 ref、default、 fixed、 form、maxOccurs、minOccurs、 abstract、block、final	simpleType、complexType、key、keyref、unique	schema、all、sequence、choice

续表

元素名	说 明	主要属性	内容	父元素
attribute	定义一个属性	id、name、type、ref、default、fixed、form、use、any attributes	annotation、simpleType	schema、complexType、attributeGroup、restriction、simpleContent、complexContent、extension
group	定义在复杂类型定义中使用的元素组	id、name、ref、minOccurs、maxOccurs	annotation、all、choice、sequence	schema、choice、sequence、complexType、restriction、complexContent、extension
attributeGroup	定义在复杂类型定义中使用的属性组	id、name、ref	annotation、attribute、attributeGroup、anyAttribute	schema、attributeGroup、complexType、restriction、simpleContent、complexContent、extension
key	指定属性或元素值(或一组值)必须是指定范围内的键	id、name	annotation、selector、field	element
keyref	规定属性或元素值(或一组值)对应指定的 key 或 unique 元素的值	id、name、refer	annotation、selector、field	element
unique	指定属性或元素值(或者属性或元素值的组合)在指定范围内必须是唯一的	id、name	annotation、field、selector	element
field	规定 XPath 表达式,该表达式指定用来定义标识约束(unique、key 和 keyref 元素)的值	id、XPath、xml:lang	annotation	key、keyref、unique
selector	指定 XML 路径语言(XPath)表达式,该表达式为标识约束选择一组元素(unique、key 和 keyref 元素)	id、XPath	annotation	key、keyref、unique

在表 3-14 中，我们只是列出了每个元素的属性，因为本书篇幅的限制，对于属性的功能、取值没有详细给出，需要时可以从网上搜索或查看相应的手册。

为了更精细地定义数据的取值范围和取值模式，在 XSD 中还定义了一组内容取值约束及限定元素，来更精确地限定元素的取值。XSD 定义的标准取值约束限定（Restrictions/Facets）元素名称见表 3-15。

表 3-15 XSD 数据类型内容取值约束元素

元素名	说　明
minExclusive	内容值范围最小，但不包含此值，即＜
minInclusive	内容值范围最小，且包含此值，即＜＝
maxExclusive	内容值范围最大，但不包含此值，即＞
maxInclusive	内容值范围最大，且包含此值，即＞＝
length	元素内容的长度
minLength	元素内容的最小长度
maxLength	元素内容的最大长度
enumeration	元素内容为此元素中选一的列表
totalDigits	指定最大数字的位数
pattern	正则语言的元素内容

通过对数据类型设置内容约束，可以使数据的有效性设置更加准确。

2. XSD 内置数据类型

在 W3C XML Schema 规范中，给出了大量的标准数据类型，分为字符串数据类型、数值型数据类型、日期时间型数据类型，以及 Boolean、进制等杂项数据类型。

1）字符串相关数据类型

在 XSD 中，对于字符串数据，进行了更加精细化的类型划分，相关类型见表 3-16。

表 3-16 字符串相关数据类型

类　型　名	说　明
string	字符串，以下的数据类型均衍生于字符串数据类型
normalizedString	规格化字符串数据类型，不包含换行符（'\10'）、回车（'\13'）或制表符（'\9'）的字符串，XML 处理器会移除换行、回车以及制表符，相邻的空格字符将会被合并为一个空格字符，第一个和最后空格将被移走
Token	不包含换行符、回车或制表符、开头或结尾空格或者多个连续空格的字符串。如果存在，XML 处理器会将它们去除，中间的多个空格合并为 1 个空格
Name	Token 的衍生类型，包含合法 XML 名称的字符串，XML 名称的第一个字母必须是字母、下划线（_）或表意字符
NCName	Name 的衍生类型，不包含冒号（:）的 XML 名称。NCName 以字母或下划线（_）字符开头，后接 XML 规范中允许的任意字母、数字、重音字符、变音符号、句点（.）、连字符（-）和下划线（_）的组合
QName	限定名（qualified name），由名字空间（namespace）、前缀（prefix）、冒号（:）和一个元素名称构成
NMTOKEN	NCName 的衍生类型，类似 XML 名称，但允许所有的字符作为名称的开始字符

续表

类　型　名	说　　明
NMTOKENS	值为合法的 XML 名称的列表，包含一个或多个用空白分隔的 XML 名称记号
ID	值为唯一的 id
IDREF	值为另外一个元素的 id
IDREFS	包含合法的语言 id 的字符串，值为其他 id 的列表
ENTITY	值是一个实体
ENTITIES	值是一个实体列表

对于字符串数据类型，可以附加的类型的限定(Restriction)有 enumeration、length、maxLength、minLength、whiteSpace 和 pattern（NMTOKENS、IDREFS 以及 ENTITIES 无法使用此约束）。

2）数值型数据类型

在 XSD 中，表 3-17 中所有的数据类型均源自十进制数据类型。

表 3-17　数值型数据类型列表

类　型　名	说　　明
decimal	十进制数字，小数或整数
byte	有正负的 8 位整数(−128～127)
unsignedByte	无正负的 8 位整数($0\sim2^8-1$，即 0～255)
integer	整数值
positiveInteger	仅包含正值的整数（$1\sim2^{31}-1$）
nonPositiveInteger	仅包含非正值的整数（$-2^{31}\sim$，−2，−1，0）
negativeInteger	仅包含负值的整数（$-2^{31}\sim$，−2，−1）
nonNegativeInteger	仅包含非负值的整数（0，1，2，～ $2^{31}-1$）
int	有正负的 32 位整数($-2^{31}\sim 2^{31}-1$)
unsignedInt	无正负的 32 位整数($0\sim2^{32}-1$)
short	有正负的 16 位整数($-2^{15}\sim 2^{15}-1$)
unsignedShort	无正负的 16 位整数($0\sim2^{16}-1$)
long	有正负的 64 位整数($-2^{63}\sim 2^{63}-1$)
unsignedLong	无正负的 64 位整数($0\sim2^{64}-1$)

在计算机中，整数类型数据采用补码表示法，正数的补码等于数值本身，负数的补码为其绝对值的原码取反+1。对于一个 n 位 2 进制整数，通常是最高位为符号位，0 代表正数，1 代表负数，其余的 $n-1$ 位为数值位。因此，一个 n 位有符号的整数，可存储的最大数的最高位为 0(代表正数)，其余 $n-1$ 位全为 1，表示的十进制数为 $2^{n-1}-1$；可存储的最小数的最高位为 1(代表负数)，其余 $n-1$ 位全为 0，表示的十进制数为 -2^n-1。对于无符号数，其最高位不再是符号位，也表示数值，因此，最大的数为所有位均为 1，其取值范围是 $0\sim2^n-1$。

可与数值数据类型一同使用的限定为 enumeration、fractionDigits、maxExclusive、maxInclusive、minExclusive、minInclusive、pattern、totalDigits、whiteSpace。

3）日期时间型数据类型

在 XSD 中，日期时间型数据类型见表 3-18。

表 3-18 日期时间型数据类型

类 型 名	说 明
date	定义一个日期值，格式为 yyyy-mm-dd
time	定义一个时间值，格式为 hh:mm:ss
dateTime	定义一个日期和时间值，字符串的格式为：yyyy-mm-ddTHH:MM:SS 例如：2007-03-11T11:09:05，其中的 T 为日期和时间的分割符
gDay	定义日期的天 (DD)
gMonth	定义日期的月 (MM)
gMonthDay	定义日期的月和天 (MM-DD)
gYear	定义日期的年 (YYYY)
gYearMonth	定义日期的年和月 (YYYY-MM)
duration	定义一个时间间隔

可附加的内容约束有 minInclusive、maxInclusive、minExclusive、maxExclusive、whiteSpace、enumeration、pattern。

4) 杂项数据类型

在 XSD 中，杂项数据类型见表 3-19。

表 3-19 杂项数据类型

类 型 名	说 明
boolean	规定 true 或 false 值。合法的布尔值是 true、false、1(表示 true) 以及 0(表示 false)
anyURI	用于规定 URI，假如某个 URI 含有空格，用 %20 替换
base64Binary	表达 Base64 编码的二进制数据
hexBinary	十六进制编码的二进制数据
float	单精度 32 位浮点数
double	双精度 64 位浮点数
NOTATION	描述 XML 文档中非 XML 数据格式

可与杂项数据类型一同使用的限定为 enumeration (布尔数据类型无法使用此约束)、length (布尔数据类型无法使用此约束)、maxLength (布尔数据类型无法使用此约束)、minLength (布尔数据类型无法使用此约束) 、whiteSpace、pattern。

3.5.4 数据类型定义

元素分为简单类型元素和复杂类型元素两种，简单类型元素的值不能包含元素或属性，复杂类型元素可以产生在其他元素中嵌套元素的效果，或者为元素增加属性。无论是 MS XML Schema 还是 W3C 的 XML Schema 规范，均包含一组预定义的数据类型，除此之外，还允许用户自定义元素数据类型。

在 XML Schema 中，数据类型的定义有两种方式，一种是单独地定义数据类型，并为类型命名；另一种是将数据类型定义写在元素声明中。两种方式各有特点，应根据实际情况而定，如果一个数据类型只在一个元素中使用，在元素声明时定义类型更加简单。

1. 简单类型及其定义

简单类型元素是不含属性或其他元素的元素。在 W3C XML Schema 规范中，已经预

定义了大量的简单类型,除此之外,还允许用户自己定义简单数据类型。用户自定义简单类型的一般形式是:

```
<xs:simpleType name = "name">
  <xs:restriction base = "xs:datatypes">
      <xs:facets_element value = "value"/>
      …
  </xs:restriction>
</xs:simpleType>
```

在上述的简单数据类型定义中,<xs:simpleType>元素的 name 属性给出类型名,<restriction>元素 base 属性设置基础数据类型,如 string、float、double 和 decimal 等,也可以是用户定义的简单数据类型。除此之外,restriction 元素的子元素 facets_element 是描述数据类型的细节规则,即元素内容取值的精确约束,比如长度、范围等,见表 3-15。

【例 3-15】 定义一个简单数据类型 age,要求为 18~65 岁的整数。

分析:该类型不仅要说明基本类型,还需要给出类型的取值范围,这可通过内容约束元素来设定。类型定义如下:

```
<xs:simpleType name =  "age">
  <xs:restriction base = "xs:integer">
    <xs:minInclusive value = "18"/>
    <xs:maxInclusive value = "65"/>
  </xs:restriction>
</xs:simpleType>
```

【例 3-16】 利用上面定义的 age 类型,定义一个年龄在 18~25 岁的青年数据类型。

类型定义如下:

```
<xs:simpleType name =  "young">
  <xs:restriction base = "age ">
    <xs:minInclusive value = "18"/>
    <xs:maxInclusive value = "25"/>
  </xs:restriction>
</xs:simpleType>
```

【例 3-17】 定义枚举数据类型 colorType。

类型定义如下:

```
<xs:simpleType name =  "colorType">
    <xs:restriction base = "xs:string">
      <xs:enumeration value = "red"/>
      <xs:enumeration value = "white"/>
      <xs:enumeration value = "blue"/>
      <xs:enumeration value = "black"/>
    </xs:restriction>
</xs:simpleType>
```

【例 3-18】 定义数值型数据类型,并约束取值的格式。

类型定义如下:

```
<xs:simpleType name = "productCode">
    <xs:restriction base = "xs:string">
        <xs:length value = "6" fixed = "true"/>
    </xs:restriction>
</xs:simpleType>
```

还可以用 pattern 字符串设置正则语言模板字符串：

```
<xs:simpleType name = "productCode">
    <xs:restriction base = "xs:string">
        <xs:pattern value = "\d{2} - \d{3} - \d{7}"/>
    </xs:restriction>
</xs:simpleType>
```

2. 复杂类型及其定义

在 XML Schema 中，一个元素如果包含属性或别的元素，这样的元素为复杂类型元素。复杂类型元素又称为容器元素，声明一个复杂类型元素需要定义复杂类型。在 W3C XML Schema 中，复杂类型的定义的一般形式为：

```
<xs:complexType name = "name" mixed = "false|true">
    <xs:子元素指示器>
        <xs:element name = "element - name" type = "element - type"/>
        <xs:element name = "element - name" type = "element - type" />
    …
    </xs:子元素指示器>
</xs:complexType>
```

其中，name 属性表示数据类型的名称，mixed 属性说明此元素是否标记内容，默认值为 false，表示只声明 XML 元素。当 mixed 的值为 true 时，表示声明 XML 元素所标记的内容中，除了 XML 元素外，还包含文字内容，参见例 3-23。

子元属类型指示器给出了子元素之间在容器中的关系，说明见表 3-20。

表 3-20　XSD 指示器元素及功能

指示器分类	指示器元素	说　明
Order 指示器用于定义元素的顺序	<all>	规定子元素可以按照任意顺序出现，且每个子元素必须只出现一次。当使用<all>指示器时，可以把<minOccurs>设置为 0 或者 1，而只能把<maxOccurs>指示器设置为 1
	<choice>	规定可出现某个子元素或者可出现另外一个子元素
	<sequence>	规定子元素必须按照特定的顺序出现
Occurrence 指示器用于定义元素出现的频率	<maxOccurs>	规定某个元素可出现的最大次数，如需使某个元素的出现次数不受限制，使用 maxOccurs="unbounded"
	<minOccurs>	规定某个元素能够出现的最小次数
Group 指示器用于定义相关元素	<Group>	元素组
	<attributeGroup>	属性组

续表

指示器分类	指示器元素	说　　明
Content 指示器用于指定数据内容	<simpleContent>	没有 XML 元素，只有属性、内容或 simpleType 内容
	<complexContent>	只拥有 XML 元素或空元素

【例 3-19】 定义复杂数据类型 person，包含姓名、性别、出生日期。然后扩展 person 类型，定义一个新的数据类型 personinfo，包含地址等信息。

类型定义如下：

```
<xs:complexType name = "person">
    <xs:sequence>
        <xs:element name = "name" type = "xs:string"/>
        <xs:element name = "sex">
               <xs:simpleType>
                 <xs:restriction base = "xs:string">
                    <xs:pattern value = "male|female"/>
                 </xs:restriction>
               </xs:simpleType>
        </xs:element>
        <xs:element name = "birthday" type = "xs:date"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name = "personinfo">
        <xs:complexContent>
            <xs:extension base = "person">
                <xs:sequence>
                    <xs:element name = "address" type = "xs:string"/>
                    <xs:element name = "city" type = "xs:string"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
</xs:complexType>
```

在对数据类型进步扩展时，<extension>元素不能直接在<complexType>元素内，必须通过<complexContent>元素。此外，对于一个复杂数据类型，还可以在最后一个元素的后面添加<xs:any minOccurs="0"/>来进行扩展。但是，此时，该类型将不能设置为<extension>元素的 base 属性。

【例 3-20】 元素组的定义及应用。

示例代码如下：

```
<xs:group name = "persongroup">
  <xs:sequence>
    <xs:element name = "name" type = "xs:string"/>
    <xs:element name = "sex" type = "xs:string"/>
    <xs:element name = "birthday" type = "xs:date"/>
  </xs:sequence>
</xs:group>
<xs:complexType name = "personinfo">
```

```
  <xs:sequence>
    <xs:group ref="persongroup"/>
    <xs:element name="address" type="xs:string"/>
    <xs:element name="city" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:element name="person" type="personinfo"/>
```

3.5.5　声明元素

声明元素就是定义元素的名字和内容模型。在 XML Schema 中,元素的内容模型由其类型定义,定义一个元素,即声明一个元素名称及给定该元素的取值类型,在 XML 文档中实例元素的值必须符合模式中定义的类型。

在 XML Schema 中,定义 XML 元素的一般形式是:

```
<xs:element name="elementname" type="datatype" default="value" fixed="value" />
```

其中,elementname 为要定义的元素的元素名,datatype 声明该元素取值的数据类型,数据类型分为简单类型和复合类型两种,简单类型元素的值不能包含元素或属性,复合类型元素可以产生在其他元素中嵌套元素的效果,或者为元素增加属性。default 属性给定默认值,fixed 属性指定固定值。

除了上述通过元素类型声明一个元素外,还可以将数据类型的定义和元素的声明进行合并,来声明一个元素。一般形式是:

```
<xs:element name="elementname">
    元素数据类型定义
</xs:element>
```

【例 3-21】　用两种不同形式声明一个元素<car>,元素内容为 Audi、BWM 或 Buick。示例代码如下。

形式一:

```
<xs:element name="car" type="carType" />
<xs:simpleType name="carType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="Audi"/>
        <xs:enumeration value="Buick"/>
        <xs:enumeration value="BMW"/>
    </xs:restriction>
</xs:simpleType>
```

形式二:

```
<xs:element name="car">
<xs:simpleType>
    <xs:restriction base="xs:string">
        <xs:enumeration value="Audi"/>
        <xs:enumeration value=" Buick"/>
        <xs:enumeration value="BMW"/>
```

```
        </xs:restriction>
    </xs:simpleType>
    </xs:element>
```

【例 3-22】 声明一个元素<password>,其取值为 6～8 个字母、数字字符构成的字符串。

示例代码如下:

```
<xs:element name = "password">
<xs:simpleType>
    <xs:restriction base = "xs:string">
        <xs:minLength value = "5"/>
        <xs:maxLength value = "8"/>
        <xs:pattern value = "[a-zA-Z0-9]{6,8}"/>
    </xs:restriction>
</xs:simpleType>
</xs:element>
```

【例 3-23】 复合复杂数据类型的定义及应用举例。

类型定义如下:

```
<xs:element name = "letter" type = "lettertype"/>
<xs:complexType name = "lettertype" mixed = "true">
    <xs:sequence>
        <xs:element name = "name" type = "xs:string"/>
        <xs:element name = "orderid" type = "xs:positiveInteger"/>
        <xs:element name = "shipdate" type = "xs:date"/>
    </xs:sequence>
</xs:complexType>
```

在定义上述复杂数据类型和元素后,可以书写下面的 XML 内容:

```
<letter> Dear Mr.< name > John Smith </name >. Your order < orderid > 1032 </orderid > will be
shipped on < shipdate > 2011-07-11 </shipdate>. </letter>
```

介绍完 XML Schema 中声明元素的方法后,下面通过一个例子来比较一下 DTD 和 XML Schema 的不同。假定有一个简单的 XML 文档内容如下:

```
<书本>
    <书名>丁丁历险记</书名>
    <作者> Georges Remi </作者>
</书本>
```

如果用 DTD 的形式来定义该 XML 文档结构,DTD 定义如下:

```
<!ELEMENT 书本 (书名, 作者)>
<!ELEMENT 书名 (#PCDATA)>
<!ELEMENT 作者 (#PCDATA)>
```

如果用 XML Schema 形式来定义 XML 文档结构,则 XML Schema 定义如下:

```
<xs:element name = "书本" type = "书本类型"/>
<xs:complexType name = "书本类型">
```

```
    <xs:all>
        <xs:element name="书名" type="string"/>
        <xs:element name="作者" type="string"/>
    </xs:sequence>
</xs:complexType>
```

其中,元素<书本>的取值类型为用户定义的一个复杂数据类型,因此,<书本>元素为复杂类型元素。

3.5.6 声明元素属性

复杂元素可以拥有子元素和元素属性,简单类型没有元素属性,因此,元素属性的声明通常是在复杂数据类型的定义中实现的。声明元素属性的一般形式是:

```
<xs:attribute name="name" type="datatype" default= "defaultvalue" use= "required|no"/>
```

其中,name 为属性名,type 是指属性的数据类型,default-vaue 是属性的默认值,use 设置该属性是否为必选属性。

【例 3-24】 声明一个空元素<product>,拥有一个长度为 6 个数字字符的编码。

示例代码如下:

```
<xs:simpleType name="idtype">
    <xs:restriction base="xs:string">
        <xs:pattern value="[0-9]{6}"/>
    </xs:restriction>
</xs:simpleType>
<xs:complexType name="prodtype">
    <xs:attribute name="prodid" type=" idtype"/>
</xs:complexType>
<xs:element name="product" type="prodtype"/>
```

【例 3-25】 声明一个复杂元素<bookorder>,拥有两个子元素<bookname>和<customer>以及一个 orderid 属性,规定一个订单可以订多本书,orderid 为长度为 6 个字符的字符串。

要实现上述要求,可分成三个基本步骤,如下。

(1) 定义简单数据类型,进行内容取值约束和限定:

```
<xs:simpleType name="orderidtype">
   <xs:restriction base="xs:string">
       <xs:pattern value="[0-9]{6}"/>
   </xs:restriction>
</xs:simpleType>
```

(2) 定义元素属性:

```
<xs:complexType name="bookordertype">
   <xs: sequence>
       <xs:element name="customer" type="xs:string"/>
       <xs:element name="bookname" type="xs:string" maxOccurs="unbounded"/>
   </xs: sequence>
```

```
    <xs:attribute name = "orderid" type = "xs:string" use = "required"/>
    <xs:anyAttribute/>
</xs:complexType>
```

(3) 声明元素：

```
<xs:element name = "bookorder" type = "bookordertype"/>
```

在元素属性的声明中，使用了<xs:anyAttribute/>，可以使创作者在元素中添加orderid以外的任意数量的属性，但是这些属性必须通过Schema声明，例如，我们在Schema中包含属性note的声明：

```
<xs:attribute name = "note">
    <xs:simpleType>
        <xs:restriction base = "xs:string">
            <xs:pattern value = "普通|加急"/>
        </xs:restriction>
    </xs:simpleType>
</xs:attribute>
```

可以将属性应用到<bookorder>元素中，对于下列XML内容：

```
<bookorder orderid = "201201" note = "加急" status = "OK"><customer>Hao</customer>订购图书
清单<bookname>Web 技术导论(第 3 版)</bookname></bookorder>
```

其中，note属性是有效的，虽然未在<bookorder>元素中声明，但status属性无效，因为在schema中未声明该属性。上述内容看起来不够自然，可以将其关系用层次缩进来表达，形式如下：

```
<bookorder orderid = "201201" note = "加急" status = "OK">
    <customer>Hao</customer>订购图书清单
    <bookname>Web 技术导论(第 3 版)</bookname>
</bookorder>
```

3.5.7 将架构应用到XML文档

用户可以在一个XML文档的内部应用一个架构，从而使用架构中声明的元素或定义的数据类型。同时，也可以对XML文档数据进行有效性验证。在XML文档中应用架构，需要在该文档的根元素中声明xmlns，一般形式是：

```
<rootelement xmlns = "Schema - URI XSD - file">
```

其中，rootelement为用户希望应用架构的XML文档的根元素。Schema-URI为统一资源标识符，是要附加的架构的名称，如果文档中需要应用多个Schema，应为它们的命名空间命名，以便在使用不同的Schema中的元素时，不至于出现元素重名。XSD-file给定包含Schema定义的xsd文件名。

下面设计一个有关家庭的模式文档family. xsd，并利用该Schema完成一个XML文档Myfamily. xml数据的验证。

代码清单：模式文件family. xsd

```
<?xml version = "1.0" encoding = "gb2312"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema"
            elementFormDefault = "qualified">
<xs:complexType name = "familytype">
    <xs:sequence>
        <xs:element name = "father" type = "xs:string" minOccurs = "0"/>
        <xs:element name = "mother" type = "xs:string" minOccurs = "0"/>
        <xs:element name = "boy" type = "xs:string" minOccurs = "0" maxOccurs = "5"/>
        <xs:element name = "girl" type = "xs:string" minOccurs = "0" maxOccurs = "5"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name = "hometype">
  <xs:all>
     <xs:element name = "house" type = "xs:string" minOccurs = "1"/>
  </xs:all>
</xs:complexType>

<xs:element name = "familys">
    <xs:complexType>
       <xs:sequence>
          <xs:element name = "family" type = "familytype" maxOccurs = "unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name = "family" type = "familytype"/>
<xs:element name = "home" type = "hometype"/>
</xs:schema>
```

下面是一个 XML 实例文档 myfamily. xml。

代码清单：myfamily. xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<familys xmlns:xsi = http://www.w3.org/2001/XMLSchema-instance
          xsi:noNamespaceSchemaLocation = "family.xsd">
<family>
    <father>Tony Smith</father>
    <boy>Linda</boy>
</family>
<family>
     <father>David Smith</father>
     <boy>mike</boy>
     <girl>Mary</girl>
     <girl>Susan</girl>
</family>
<family>
     <father>Michael Smith</father>
</family>
</familys>
```

对于上述的 XML 文档，可使用 XMLSpy 验证其有效性，详细介绍参见 3.4 节。

3.6 其他相关技术

XML 技术包含的内容很多,除了 XML 元语言规范外,XML 文档的有效性、XML 数据处理、XML 数据转换、XML 数据显示等都是 XML 技术非常重要的内容。了解这些技术及它们的用途、相互之间的关系,对于理解 XML 技术至关重要。

3.6.1 XML 文档对象模型

XML 文档对象模型(Document Object Model,DOM)是 W3C 制定的针对 XML 解析器的标准接口规范,是 XML 文档的应用程序接口。在应用程序(例如浏览器)中,基于 DOM 的 XML 解析器依据 XML 的文档结构,将一个 XML 文档转换成一棵节点树(通常称 DOM 树),应用程序通过对这个对象模型的操作,间接地实现对 XML 文档数据的操作。

由于 XML 本质上就是一种分层结构,用 DOM 树对 XML 文档进行描述的方法是相当有效的。DOM 树所提供的随机访问方式给应用程序的开发也带来了很大的灵活性,它可以任意地控制整个 XML 文档中的内容。然而,由于 DOM 分析器把整个 XML 文档转化成 DOM 树放在了内存中,因此,当文档比较大或者结构比较复杂时,对内存的需求就比较高。而且,对于结构复杂的树的遍历也是一项耗时的操作。所以,DOM 解析器对机器性能的要求较高,实现效果不十分理想。但是,由于 DOM 解析器所采用的树形结构的思想与 XML 文档的结构相吻合,又可以进行随机访问,因此,DOM 解析器具有广泛的使用价值。

1. XML DOM 的组成

对于 XML 应用开发来说,DOM 就是一个对象化的 XML 数据接口,一个与语言无关、与平台无关的标准接口规范。它定义了 HTML 文档和 XML 文档的逻辑结构,给出了一种访问和处理 HTML 文档和 XML 文档的方法。利用 DOM,程序开发人员可以动态地创建文档,遍历文档结构,添加、修改、删除文档内容,改变文档的显示方式,等等。可以这样说,文档代表的是数据,而 DOM 则代表如何去处理这些数据。无论是在浏览器中,在服务器上,还是在客户端,只要用到 XML,都可能用到 DOM。

作为 W3C 的标准接口规范,目前,DOM 由三部分组成,即核心(core)、HTML 和 XML。核心部分是结构化文档比较底层对象的集合,定义了一组对象,用于表达 HTML 和 XML 文档中的数据。HTML 接口和 XML 接口两部分则是专为操作具体的 HTML 文档和 XML 文档所提供的高级接口,使对这两类文件的操作更加方便。

2. 创建 DOM 文档对象和加载 XML

通过 DOM 访问 XML 数据通常有两种方法:①使用 C、C++或者 Visual Basic 等访问 XML 数据,此时用户必须安装一些特殊的头文件和库;②使用 HTML 和脚本语言 JavaScript、VBScript 等通过 DOM 实现 XML 数据的访问。这里我们只看后者的实现方法。

在 HTML 中访问 XML,首先要创建文档对象。在 JavaScript 中,可通过以下语句

完成：

```
var obj = new ActiveXObject("Microsoft.XMLDOM")
```

上述语句在内存中创建一个空对象 obj，要加载一个 XML 文档，需通过 obj 对象的 load 方法完成：

```
Obj.async = false
Obj.load("XML 文档的 URL")
```

这里我们给 async 属性赋值为 false，表明只有当文档下载完毕，控制才返回给调用进程。load 方法告诉分析器加载指定名字的 XML 文档。XML 文档被加载后，就在内存中形成了一棵 DOM 树。

例如，books.xml 文档内容如下：

```
<?xml version = "1.0" encoding = "gb2312" ?>
<books>
<book status = "已售完">
        <title>Web 技术导论</title>
        <author>HaoXingwei</author>
</book>
<book status = "热卖中">
        <title>Web 开发技术</title>
         <author>HaoXingwei</author>
</book>
</books>
```

形成的 DOM 树如图 3-11 所示。

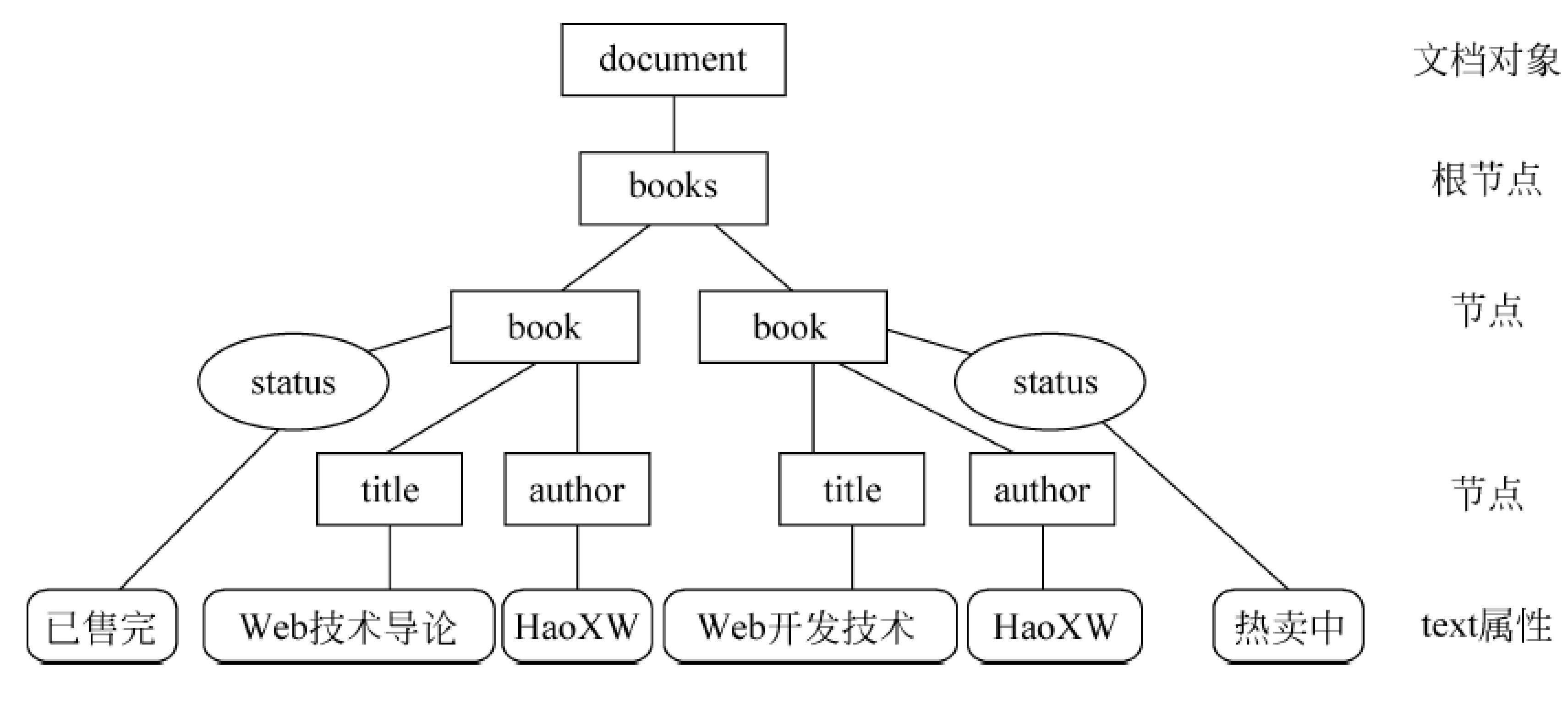


图 3-11　XML DOM 树示意

从图 3-11 所示的 DOM 树中可以看到，在文档对象中，包含了一个与 XML 文档相一致的树。树的根节点对应于 XML 的根元素，其他节点对应于 XML 文档中不同层次上的元素。所有节点均为 Node 类型对象。

3. XML DOM 对象

文档对象模型利用对象将 XML/HTML 文档模型化，这些模型不仅描述了文档的结

构,还定义了模型中对象的行为。在 DOM 树中,每一个节点不是数据结构,而是对象,对象中包含属性和方法。在 DOM 接口规范中,有 4 个基本的接口:Document、Node、NodeList 和 NamedNodeMap。

1) 文档对象(Document)

文档对象是从 Node 接口继承过来的,代表整个 XML/HTML 文档,是整棵文档树的根,是所有数据、所有其他组件,比如注释和处理指令节点的总的容器,提供了对文档中的数据进行访问和操作的入口。

(1) Document 对象常用属性如下。

① documentElement:Element 类型的只读属性,返回文档的根节点对象。

② async:逻辑类型的属性,指定是否允许同步下载文件。

③ parseError:返回解析错误对象。

(2) Document 对象常用方法如下。

① load(pathname):把文件加载到文档对象。

② loadMXL(string):加载 XML 文档或段。

③ createAttribute(name):创建属性方法。

④ createNode(Type,Name,namespaceURI):创建一个类型为 Type、名称为 Name,并且使用 namespaceURI 作为名称空间的节点。

2) 节点对象(Node)

XML/HTML 中的每一个元素都对应 DOM 树中的一个节点对象。Node 接口是其他大多数接口的父类,如 Documet、Element、Attribute、Text、Comment 等接口都是从 Node 接口继承过来的。

Node 对象常用属性见表 3-21。

表 3-21 Node 对象常用属性

属性名	意 义	属性名	意 义
atrribute	节点的属性集	nextSibling	当前节点的下一个兄弟节点
childNodes	当前节点所有子节点的 nodeList	nodeName	节点名
dataType	设置或获得节点数据的类型	nodeType	节点类型
firstChild	当前节点的首子节点	parentNode	父节点对象
lastChild	当前节点的末子节点	text	设置或获得节点文本
nameSpace	返回名称空间的 URI	xml	返回指定节点的 xml 表示

Node 对象常用方法见表 3-22。

表 3-22 Node 对象常用方法

方法名	功 能	方法名	功 能
appendChild	添加新节点	removeChild	删除子节点
cloneNode	克隆节点	replaceChild	替换节点
hasChildNodes	当前节点是否有子节点	selectNodes	选择节点列表及其后代
insertBefore	插入节点	selectSingleNode	选择节点及其后代
parsed	节点是否被解析		

3）节点列表对象（NodeList）

NodeList 接口是一个节点的集合，它包含某个节点中的所有子节点。在 DOM 中，NodeList 的对象是实时更新的，对文档的改变，会直接反映到相关的 NodeList 对象中。例如，如果通过 DOM 获得一个 NodeList 对象，该对象中包含某个 Element 节点的所有子节点的集合，当通过 DOM 对 Element 节点进行添加、删除等操作时，这些改变将会自动地反映到 NodeList 对象中，而不需 DOM 应用程序再做其他额外的操作。

NodeList 是一个节点对象列表，只有一个 Length 属性，其值为对象中包含节点的个数。NodeList 中的每个 item 可以通过一个索引来访问，该索引值从 0 开始。主要的方法有 item(Index)、nextNode()和 reset()，详细介绍略。

4）名字空间对象（NamedNodeMap）

NamedNodeMap 接口也是一个节点的集合，通过该接口，可以建立节点名和节点之间的一一映射关系，从而利用节点名可以直接访问特定的节点。和 NodeList 对象不同，NamedNodeMap 所包含的节点集中的节点是无序的。尽管这些节点也可以通过索引来进行访问，但这只是提供了枚举 NamedNodeMap 中所包含节点的一种简单方法，并不表明在 DOM 规范中为 NamedNodeMap 中的节点规定了一种排列顺序。

【例 3-26】 设有一个 XML 文档 books. xml，记录了有关图书数据。要求写一个 HTML 文档，通过 DOM 模型将 books. xml 中各本书的书名加到一个列表控件中，单击列表控件中的书名，显示该书数据。

分析：这是一个利用 HTML 文档显示 XML 文档数据的问题，也很好地反映了 HTML 用于数据展示和 XML 用于数据表达的思想定位。

代码清单 1：chinesebooks. xml 文档内容

```
<?xml version = "1.0" encoding = "gb2312" ?>
<中国古典名著>
 <图书>
  <书名>三国演义</书名>
  <作者>罗贯中</作者>
</图书>
<图书>
  <书名>西游记</书名>
  <作者>吴承恩</作者>
</图书>
<图书>
  <书名>红楼梦</书名>
  <作者>曹雪芹</作者>
  </图书>
 <图书>
  <书名>水浒传</书名>
  <作者>施耐庵</作者>
</图书>
 </中国古典名著>
```

代码清单 2：chinesebook. htm 文档内容

```
< html >
```

```
<head>
<script language="JavaScript">
function createlist()
{
   var str="<select size='4' id='D1' onChange='selectbook(D1.selectedIndex)'>";
   var mybook=new ActiveXObject("Microsoft.XMLDOM");
   mybook.async=false;
   mybook.load("chinesebook.xml");
   //返回文档的根节点对象
   root=mybook.documentElement;
   for (var i=0; i<root.childNodes.length; i++)
   {
         booknode=root.childNodes.item(i);
         a1=booknode.childNodes.item(0).text;
         str +=  "<option>" + a1 + "</option>";
   }
   str +="</select>";
   booklist.innerHTML=str;
}
function selectbook(i)
{
  booknode=root.childNodes.item(i);
  booktitle.innerText  =booknode.childNodes.item(0).text;
  bookauthor.innerText=booknode.childNodes.item(1).text;
}
</script>
</head>

<body>
<h1 align="center">中国古典名著</h1>
<table border="1" width="400" bgcolor="#CCFFCC">
<tr height="30">
  <td width="100" align="center">书名</td>
  <td align="center">图书信息</td>
</tr>
<tr>
   <td id="booklist" align="center" ></td>
   <td  align="left">
        <p>书名:<span id="booktitle"></span></p>
        <p>作者:<span id="bookauthor"></span></p>
  </td>
</tr>
</table>
<script>
  createlist();
</script>
</body>
</html>
```

在上述 chinesebook.htm 代码中,通过脚本程序生成一个下拉列表,当单击其中列表项时,修改右侧单元格的图书信息。文档 chinesebook.htm 在 IE 中的显示结果如图 3-12 所示。

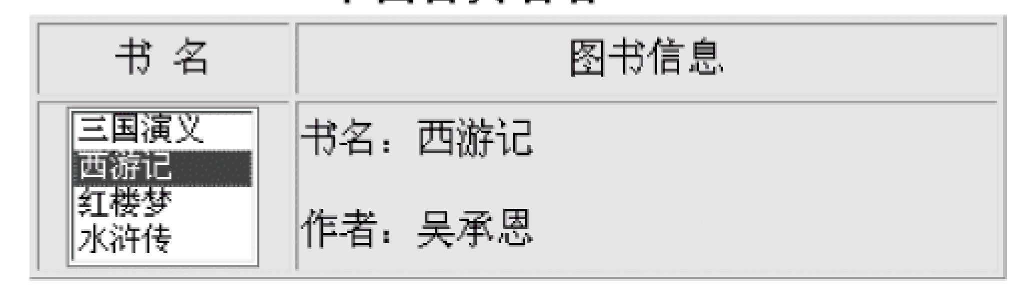


图 3-12　应用 DOM 对象示例

不是所有的浏览器均支持 Microsoft.XMLDOM，Firefox 浏览器将不能显示上述页面。通过上面的这个例子，可以看出 xml 文档是对数据的一种描述，是存储数据的一种形式，而 html 文档的重点在于展示数据。理解这一点对于理解 xml 是非常重要的。

3.6.2　可扩展样式语言

W3C 给出了两种样式单语言的推荐标准，一种是层叠样式单(CSS)，另一种是可扩展样式单语言(eXtensible Stylesheet Language，XSL)。样式单(Style Sheet)是一种专门描述结构文档表现方式的文档，它既可以描述这些文档如何在屏幕上显示，也可以描述它们的打印效果，甚至声音效果。使用样式单的目的就是要保证文档内容和文档显示的彻底分离。CSS 主要应用在 HTML 的页面制作，而可扩展样式语言(eXtensible Style Language，XSL)则是用于描述 XML 文档样式的语言。

1. XSL 简介

在 Internet 中，XML 已经成为不同应用之间的数据交换标准，从根本上解决了应用系统间的信息交换，保证了数据的平台无关性。为了使数据适合不同的应用程序，我们必须能够将一种数据格式转换为另一种数据格式，比如，需要的格式可能是一个文本文件、一个 SQL 语句、一个 HTTP 信息、一定顺序的数据调用等。

此外，对于 XML 文档数据，可能还需要将 XML 数据以不同的方式进行显示。在 XML 技术中，可以和 HTML 一样，定义样式表(.css)文件，在 XML 文档的序言中增加声明显示该 XML 文档要使用的样式表，处理指令为“<? xml-stylesheet type="text/css" href="mycssfile.css"? >”，然后通过在 XML 元素中使用 class 属性和 id 属性来定制元素的显示。但是，必须要记住的是：XML 技术的核心是将数据和数据的显示分离，任何在 XML 文档内容中包含显示信息的 XML 文档都认为是含有 bug 的，因此，这样的设计不符合 XML 的思想。XML 技术的基本规则是，XML 文档存储数据本身，对数据的显示则通过 XML 数据的应用程序来完成。

基于上述两个方面的原因，W3C 颁布了可扩展样式语言(XSL)规范，用于 XML 文档的转换和显示。XSL 在转换 XML 文档时分为两个过程：首先转换文档结构，然后将文档格式化输出。这两步可以分离开来并单独处理，因此 XSL 在发展过程中逐渐分为 XSLT (XSL Transformations)和 XSL-FO(XSL Formatting Objects)两种分支语言。XSLT 用来实现 XML 文档结构的转换，其中，将 XML 文档转换为 HTML 文档是目前 XSLT 最主要的功能。XSL-FO 的作用则类似 CSS 在 HTML 中的作用。

2. XSLT 样式文档基本结构

XSL 样式文档用于处理 XML 文档内容的格式化显示,可以说 XML+XSL 就可以达到 HTML 的显示效果,但 XSL 对数据的显示更灵活,对于同一个 XML 文档,可以编写不同的 XSL,来得到不同的显示。

XSL 文档本身是一个 XML 文档,遵守 XML 的语法规则,是 XML 的一种具体应用。XSL 样式文档基本结构如下:

```
<?xml version = "1.0" encoding = "gb2312"?>
<xsl:stylesheet version = "2.0"
                xmlns:xsl = http://www.w3.org/1999/XSL/Transform
                xmlns:xs = "http://www.w3.org/2001/XMLSchema"
                xmlns:fn = "http://www.w3.org/2005/xpath-functions">
    <xsl:output method = "xml" version = "1.0" encoding = " gb2312" indent = "yes"/>
    模板定义 1
    模板定义 2
 …
</xsl:stylesheet>
```

第一行为 XML 处理指令,告诉 XML 解析器文档为 XML 文档。<stylesheet>元素是 XML 文档的根元素,version 属性指示该 XSL 文档遵从哪一个版本的 XSL 标准,xmlns:xsl 指示了 XSL 的命名空间。<output>元素设置 XSL 的输出文档类型,其中属性 method="xml"表明输出为 XML 文档,如果要输出 HTML 文档,则应设置 method="html"。如果文档中不出现 xsl:output,将默认输出为 XML 文档,但如果在匹配模板时使用了<html>标记,则输出为 html 文档。

XSL 文档的主要内容是一系列的模板,通过一系列模板定义来描述 XML 文档的显示格式。模板通常是由 html 标记和 XSL 元素来描述的,以实现 XML 文档内容的定位和处理。XSL 由多个模板构成,其中匹配 XML 根元素的模板为默认模板,通过调用相应的模板,完成 XML 文档内容的输出。

XSL 样式文档文件扩展名为.xsl,要使用 xsl 显示一个 XML 文档,在 XML 文档声明的后面需要增加一条处理指令,声明用于显示该 XML 文档的 xsl 程序。一般形式如下:

```
<?xml-stylesheet type = "text/xsl" href = "xslfile.xsl"?>
```

其中,xslfile.xsl 是用于该 xml 文档输出的 XSLT 文件。

3. XSLT 元素

在 XSLT 规范中,定义了大量的 XSLT 元素和 XSLT 函数,用于对 XML 文档的转换,这些常用的 XML 元素见表 3-23。

在 XSL 样式文档中,元素<choose>、<when>和<otherwise>总是一起使用,构成程序设计中的多分支选择逻辑。而<element>和<attribute>元素则可以通过<if>元素动态地在输出中添加元素或添加元素属性。下面举例说明 XSL 文档中 XSL 元素的应用。

表 3-23 XSL 处理元素列表

XSL 元素	元素属性	说明
stylesheet		声明文档为样式表文档，XSLT 文档如果由模板组成，任何一个 XSLT 文件至少包含一个模板
template	name，模板名称，可选属性 match，模板所匹配的元素名字，文档的根为"/"，对应模板为默认模板 priority，模板的优先级编号(0～9) mode，模式值，允许多次处理某个元素，每次产生不同的结果	定义一个模板，每个模板负责处理 XML 文档内某一类型的标记。模板元素标记的内容将被输出，通常是 HTML 代码
apply-templates	select，被选择的节点名或节点集 mode，同 template 元素	XSL 处理器选择一个基于某个匹配的模板
call-template	name，被调用的模板的 name 属性值	调用模板名所指定的模板
value-of	select，被选择的节点名	输出被选择节点的值或属性值
copy		复制指定节点下的内容到输出文档
if	test，测试用的 boolean 表达式	简单条件判断，当条件值为 True 时执行该元素内容，否则不执行该元素内容。如果存在 script 属性，则可以用脚本语言表达式为测试条件
choose		与 xsl:when 和 xsl:otherwise 元素一起，提供多条件判断。类似于 switch 语句
when	test，同＜if＞元素	在一个 xsl:choose 元素内提供单一条件判断
otherwise		为一个 xsl:choose 元素提供默认处理
for-each	select，确定要循环访问的节点集 order-by 设方式，在节点名前用"＋"，表示按降序排列，"－"表示按升序排列	对 select 属性指定的节点集合中的节点循环执行处理
element	name，元素名称，通常是一个 HTML 标记 namespace，元素的命名空间 URI use-attribute-sets，通过空白分隔的属性集列表	输出一个 HTML 元素
attribute	name，必选项，为要创建的标记属性的名称 namespace，同 element 元素	标记添加属性，并设置属性值
comment		在输出中创建一个注释节点
include	href，标识要包含的 XSLT 文件的统一资源标识符（URI）引用	在一个 XSLT 文件中包含另一个 XSLT 文件

设有一 xml 文档 bookslist.xml，代码如下：

```
<?xml version = "1.0" encoding = "gb2312"?>
<?xml - stylesheet type = "text/xsl" href = "bookslist.xsl"?>
<collection>
  <bookitem>
```

```
        <title>The Adventures of Tom Sawyer</title>
        <author>Twain,Mark</author>
        <isbn>978-7-12345678-a</isbn>
    </bookitem>
    <bookitem>
        <title>The Little Match-Seller</title>
        <author>Andersen, Hans C.</author>
        <isbn>978-7-12345678-b</isbn>
    </bookitem>
</collection>
```

在上述 XML 文档中,第二行处理指令指示要引用的 XSL 文档。

【例 3-27】 写一个 XSLT 文档 bookslist. xsl,将 bookslist. xml 数据按照表格形式输出。

代码清单:bookslist. xsl

```
<?xml version="1.0" encoding="gb2312"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/TR/WD-xsl">
<xsl:template match="/">
<html>
<head>
<style>
    .title {font-size:15pt; font-weight:bold; color:blue }
    .name {color:red}
</style>
</head>
<body>
     <xsl:apply-templates select="/collection"/>
</body>
</html>
</xsl:template>

<xsl:template match="/collection">
<p class="title" align="center">中文图书清单</p>
<table border="1" align="center">
<thead>
    <td align="center" width="250"><b>书名</b></td>
    <td align="center" width="150"><b>作者</b></td>
    <td align="center" width="150"><b>书号</b></td>
</thead>
<xsl:for-each select="/collection/bookitem" order-by="-isbn">
<tr>
    <td><xsl:value-of select="title"/></td>
    <td><xsl:value-of select="author"/></td>
    <td align="center"><xsl:value-of select="isbn"/></td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

在 bookslist.xsl 中，我们定义了两个模板，第一个模板为默认模板，输出了 HTML 文档的框架，并调用了第二个模板，图书清单表格的输出是通过第二个模板完成的。

输出结果如图 3-13 所示。



图 3-13 XML 文档的 XSLT 输出界面

【例 3-28】 有一个描述网络课程知识单元的 XML 文档，要求编写一个 XSLT 文档，输出该 xml 文档内容。

分析：对 XML 文档内容的输出和显示有多种方法，前面我们看到了使用 DOM 的方法，也可以使用 css，但这两种输出前者与浏览器有关，css 则不符合 xml 数据与显示彻底分离的思想，正确的输出就是 XSL。

代码清单 1：知识单元 XML 文档内容清单(ku-osi.xml)。

```
<?xml version = "1.0" encoding = "gb2312" ?>
<?xml - stylesheet type = "text/xsl" href = "ku - osi.xsl" ?>
<ku>
<title>2.1 The OSI model</title>
<LearningObjectives>
  <objectiveitem>(1)了解 OSI 七层模型</objectiveitem>
  <objectiveitem>(2)掌握各层名称及其功能</objectiveitem>
  <objectiveitem>(3)理解数据封装过程</objectiveitem>
</LearningObjectives>

<kplist>
  <kpitem>
    <kptitle>OSI 模型</kptitle>
    <loitem filename = "osi.swf">OSI 模型</loitem>
    <loitem filename = "osiencapsulation.swf">数据封装过程</loitem>
  </kpitem>

  <kpitem>
    <kptitle>数据封装</kptitle>
    <loitem filename = "osiencapsulation.swf">数据封装过程</loitem>
```

```
  </kpitem>
</kplist>

<contents>
20 世纪 80 年代末 90 年代初,<kp>计算机网络</kp>迅猛发展.为了帮助网络厂商建构网络以及支持
网络互操作,<kp>ISO 组织</kp>定义了异质系统互联的七层框架体系结构标准,也称为<kp>OSI 参
考模型(OSI  Reference Model)</kp>.
  ....
</contents>

<addfilelist>
  <figitem filename = "internetclouds.jpg">图 2 - 1 互联网概念模型</figitem>
</addfilelist>

<refs>
   <refitem filename = "jsjwl.htm">计算机网络的概念和分类</refitem>
   <refitem filename = "wlsb.htm">网络设备及其功能</refitem>
</refs>
</ku>
```

下面是用于显示上述 XML 文档的 XSLT 程序清单。

代码清单 2:XSLT 文档 ku-osi.xsl 内容清单。

```
<?xml version = "1.0" encoding = "GB2312"?>
<xsl:stylesheet xmlns:xsl = "http://www.w3.org/TR/WD - xsl">

<xsl:template match = "/">
  <html>
  <body>
  <table border = "0" width = "100 % " cellspacing = "0" cellpadding = "0">
  <tr>
    <td height = "26" bgcolor = "#CC0033">
    <font color = "#FFFFFF"><xsl:value - of select = "ku/title"/></font>
    </td>
  </tr>
  </table>
  <xsl:apply - templates select = "/ku/LearningObjectives"/>
  <xsl:apply - templates select = "/ku/contents"/>
  <xsl:apply - templates select = "/ku/addfilelist"/>
  <xsl:apply - templates select = "/ku/kplist"/>
  </body>
  </html>
</xsl:template>

<xsl:template match = "/ku/LearningObjectives">
  <table border = "0" width = "100 % " cellspacing = "0" cellpadding = "0">
  <tr height = "30">
    <td><b>学习目标</b></td>
  </tr>
  <xsl:for - each select = "objectiveitem">
    <tr height = "25">
```

```
        <td style="font-size:14px;"><xsl:value-of/></td>
      </tr>
     </xsl:for-each>
    </table>
</xsl:template>

<xsl:template match="/ku/kplist">
    <hr/>
    <table>
    <tr height="30">
        <td colspan="2"><b>主要知识点</b></td>
    </tr>
    <tr>
        <td width="150">知识点</td>
        <td>学习对象</td>
    </tr>
    <xsl:for-each select="/ku/kplist/kpitem">
      <tr>
      <td><xsl:value-of select="kptitle"/></td>
      <td>
          <xsl:for-each select="loitem">
          <a>
            <xsl:attribute name="href"><xsl:value-of select="@filename"/>
            </xsl:attribute>
            <xsl:value-of/></a>
          </xsl:for-each>
     </td>
     </tr>
     </xsl:for-each>
    </table>
</xsl:template>

<xsl:template match="/ku/contents">
    <hr/>
    <table>
    <tr>
        <td><xsl:value-of select="/ku/contents"/></td>
    </tr>
    </table>
</xsl:template>

<xsl:template match="/ku/addfilelist">
     <hr/>
     <table>
      <tr height="30">
          <td colspan="10"><b>插图列表</b></td>
      </tr>
      <tr>
        <xsl:for-each select="figitem">
        <td align="center">
            <xsl:element name="img">
```

```
            <xsl:attribute name="src"><xsl:value-of select="@filename"/>
        </xsl:attribute>
          </xsl:element>
          <br/>
          <xsl:value-of/>
        </td>
        </xsl:for-each>
      </tr>
    </table>
  </xsl:template>

  </xsl:stylesheet>
```

上述 XML 文档定义了一个知识单元的结构，包括知识点，每一个知识点都对应一个或多个学习对象，每一个学习对象对应一个指向学习对象媒体文件的超链接，超链接<a>属性是通过<xsl:attribute>元素来设定的。知识单元还可以附加本知识单元的一些图、表等，对于 xml 中图片的显示，和超链接的显示不同，例子给出了另外一种方法，即通过<xsl:element name="img">来显示。

此外，对<xsl:value-of/>元素不指定参数 select，则输出当前元素的内容，如果要输出一个元素的属性值，使用"元素名/@属性"的形式，如果是当前节点，使用"@属性"即可。

当 XML 文档在浏览器中打开时，显示的页面如图 3-14 所示。

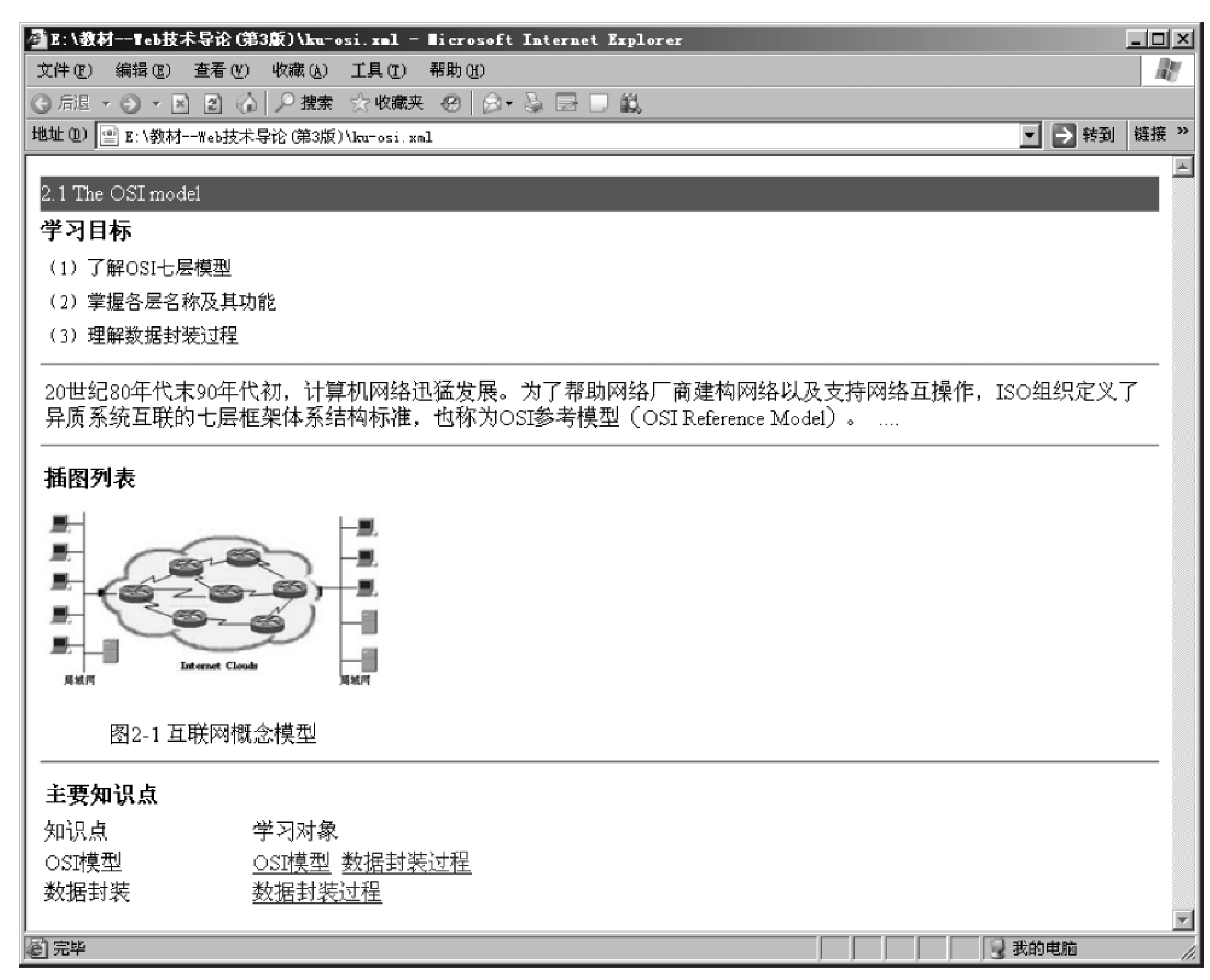


图 3-14　XML 文档使用 XSLT 的输出界面

3.6.3 XML 路径语言

XML 路径语言(XML Path Language,XPath)是用来查询和定位 XML 文档中的元素或文本的一种通用查询语言。不管是在 XSLT 还是 XPointer 规范中,都需要定位 XML 文档树中的元素、元素属性或元素区间,XPath 规范的目的就是向 XSLT 和 XPointer 共同需要的功能提供统一的语法和语义,可以说,XPath 是一种 XML 文档内容寻址语言。

XPath 将一个 XML 文档建模成一棵节点树,通过路径表达式来实现对 XML 文档内容的定位,在 XML 文档中选择节点或节点集,从而实现对 XML 文档中的元素和属性进行遍历。XPath 1.0 于 1999 年 11 月 16 日成为 W3C 推荐标准,2007 年 1 月 23 日,XPath 2.0 成为 W3C 推荐标准,很好地修复了 XPath 1.0 版中的问题。

本节将以一个简单的 XML 文档为例,介绍 XPath 中相关的基本概念及应用。

```
<?xml version = "1.0" ?>
<bookstore>
<book lang = "eng">
    <title>Aladdin and the Enchanted Lamp</title>
    <price>2.90</price>
</book>
<book lang = "eng">
    <title>Huckleberry Finn</title>
    <price>2.80</price>
</book>
</bookstore>
```

1. XPath 表达式

XPath 表达式是使用运算符和特殊字符构造的,XPath 表达式可以返回 4 种类型值,即节点集合(node set)、布尔值(Boolean)、数字和字符串。其中,XPath 最多的应用就是用于确定查询范围,即选择 XML 文档中的节点集合。

XPath 运算符和特殊字符见表 3-24。

表 3-24 XPath 运算符和特殊字符

运算符或特殊字符	说 明	示 例
/	子运算符,在左侧集合中的直接子节点元素中,选择运算符右侧的指定元素。当此路径运算符出现在模式开头时,表示选择根节点下的直接子元素	item/itemName,匹配<item>节点下的<itemName>子节点
//	递归下降,在左侧集合的任意深度子节点中搜索运算符右侧的指定元素。当此路径运算符出现在模式开头时,表示应从根节点递归下降	item/itemName,匹配<item>节点下的所有<itemName>子节点
.	取当前节点元素	.

续表

运算符或特殊字符	说　明	示　例
..	取当前节点的双亲元素节点	../itemName,父节点下的 itemName 节点
*	通配符,选择所有元素或属性	book/*,＜book＞节点下的所有子节点
@	属性名前缀	book/@isbn,＜book＞节点的 isbn 属性
@*	属性通配符(选择所有的属性)	book/@*,＜book＞节点下的所有属性
:	命名空间分隔符,将命名空间前缀与元素名或属性名分隔	
()	为运算分组,明确设置优先级	
[]	应用筛选模式或下标运算符。可以在其内指定元素或属性,也可加上额外的测试条件	book[@isbn],＜book＞节点下有属性"isbn"的节点
|	与多个节点匹配	book|title,与＜book＞或＜title＞匹配
!	应用一个信息方法至引用节点	!nodeName(),返回节点的确定名称

在介绍 XPath 应用前,先说明一下上下文节点的概念。所谓 XPath 上下文节点,就是在模板设计中,通过＜xml:template＞元素的 match 属性定位的节点,它是当前操作的节点,所有的节点定位均是基于该节点的。一般而言,最初始的上下文节点总是文档中的确定位置,如文档的根节点。在默认模板中,match="/",将上下文节点定位为文档的根。

XPath 的常用表达式说明如下。

(1) 文档根,以正斜杠(/)为前缀的表达式使用文档树的根作为上下文。需要说明的是,根节点不是根元素,它是一个抽象的节点,是根元素的父节点,用"/"表示。

例如,/bookstore,选择根元素 bookstore; bookstore/book,选择 bookstore 下的所有 book 子元素; /bookstore/*,选择 bookstore 元素下的所有子节点。

(2) 根元素,使用正斜杠后接星号(/*)的表达式使用根元素作为上下文。

例如,/*,表示查找文档的根元素。

(3) 特定元素,以元素名开头的表达式引用特定元素的查询,从当前上下文节点开始,选择当前节点的所有子节点。

例如,bookstore,选择 bookstore 元素的所有子节点。

(4) 递归下降,使用双正斜杠(//)表达式,表示从当前节点开始选择符合条件的所有节点元素。如果此运算符出现在模式的开头,上下文相对于文档的根。

例如,//book,选择文档中的所有 book 元素; //*?,? 选择文档中的所有元素; bookstore//book,选择 bookstore 下的所有 book 子孙元素。

.// 前缀则表明上下文从层次结构中当前上下文所指示的级别开始。

(5) 选择属性,//@lang,选择所有名为 lang 的属性; //title[@*],选择所有 title 元素,不管其属性如何。

(6) 多路径选择,如果需要选择多个路径,可以使用"|"操作符。例如:

//book/title|//book/price,选择所有 book 元素下的 title 和 price 子元素。

//title|//price,选择文档中的所有的 title 元素和 price 元素。

/bookstore/book/title|//price,选择/bookstore/book/下的 title 子元素和文档中的所

有 price 元素。

2. 位置路径与位置步

在 XPath 表达式中由反斜杠分开的每个部分被称为一个位置步(Location Step),换句话说,位置路径(Location Path)是由"/"分隔的位置步构成的一个表达式。位置步在目标文件中指定一个位置,通常是相对于一个已知的位置,如文件的根节点或另外一个位置步等。位置步由一个 XPath 轴关键字(axis)、节点测试(node-test)和可选谓词(predicate)构成,一般形式如下:

```
axis::node - test[predicate]
```

其中,轴是与上下文节点相对的文档的一部分,它定义了一组与当前节点有特定层次关系的节点。节点测试可以用来指示位置路径中一组合法节点的任何表达式,节点测试通过名字或类型筛选初始结果集。谓词是一个逻辑表达式。例如:

```
/child::spec/child::body[position() = 2]
```

其中,第一个"/"表示根节点;"child::spec"是第一个位置步,表示根节点的 spec 直接子元素,如果文件的根元素是 sepc,计算出来的结果应为根元素,否则计算结果为空;"child::body[position()=2]"是第二个位置步,表示上次计算出来的元素的第二个 body 直接子元素。在上面的例子中,如果文件的根元素是 spec,而且它包含两个以上的 body 子元素,返回的结果将是一个元素,否则将不返回结果。

3. 谓词

谓词用于寻找特定的节点或一个包含特定值的节点,谓词写在方括号中。例如:

/bookstore/book[1],表示选择 bookstore 元素下的第 1 个 book 子元素。

/bookstore/book[last()],选择 bookstore 元素下的最后一个 book 子元素。

/bookstore/book[position()<3],选择 bookstore 元素下开始的两个 book 子元素。

//title[@lang],选择所有的包含属性名为 lang 的 title 元素。

//title[@lang='eng'],选择所有的包含属性名为 lang 且属性值为 eng 的 title 元素。

/bookstore/book[price>2.00],选择 bookstore 元素下所有包含值大于 2.00 的 price 元素的 book 子元素。

/bookstore/book[price>2.00]/title,选择 bookstore 元素下的所有包含值大于 2.00 的 price 元素的 book 子元素下的 title 子元素。

4. XPath 轴

XPath 轴(axis)是与上下文节点相对的文档的一部分,它定义了一组与当前节点有特定层次关系的节点。各轴心及含义见表 3-25。

XPath 轴心应用于位置路径表达式中的位置步的定义,在轴心名后加"::"来确定一个位置步中的节点集。下面是一组 XPath 表达式及含义说明。

(1) ./Order,返回当前上下文中名为 Order 的所有元素。

(2) /Order,返回文档树中根下所有名为 Order 的元素。

表 3-25 XPath 轴心

轴心名称	说明
self	选择当前节点,也可以用“.”来表示
attribute	选择当前节点的属性
parent	选择当前节点的双亲
ancestor	选择当前节点的所有祖先
ancestor-or-self	选择当前节点及其所有祖先
child	选择当前节点的所有子女
descendant	选择当前节点的所有子孙
descendant-or-self	选择当前节点及其所有子孙
preceding	选择当前节点前面的节点
preceding-sibling	选择当前节点前面,与上下文节点同属于一个父节点的那些节点
following	选择上下文节点后面的元素节点
following-sibling	选择当前上下文节点后面,与上下文节点同属于一个父节点的那些节点
namespace	选择当前节点的所有名字空间节点

(3) //Order,返回从文档树的任何地方找到的所有名为 Order 的元素,不管深度或层次结构。

(4) child::book[attribute::publisher=‘张三’],返回孩子节点中出版商为张三的 book 元素节点。

(5) child::book[@publisher=‘张三’],同上。

(6) descendent::book[count(child::chapter)>5],用 count 函数检索所有 chapter 大于 5 的后代 book 节点。

(7) child::book[start-with(attribute::publisher,‘张’)],使用 start-with 函数检索所有 publisher 属性以“张”开头的 book 子节点。

【例 3-29】 设有一个 xml 文档 bookstore.xml,使用 XPath,编写一个用于输出该文档的 bookstore.xslt,并在浏览器中查看输出结果。

分析: 编写 xslt 文档,定位节点(集)及对节点和属性访问是关键,可以利用 XPath 来完成。

代码清单: bookstore.xml

```
<?xml version = "1.0" encoding = "gb2312"?>
<?xml - stylesheet type = "text/xsl" href = "bookstore.xslt"?>
<bookstore>网上书店
  <名称> World famous novels </名称>
  <图书 日期 = "2003 - 11 - 24">
     <名称> ROMEO AND JULIET </名称>
     <作者> Shakespeare,William </作者>
     <价格> 30.00 </价格>
  </图书>
  <图书 日期 = "2006 - 5 - 8">
       <名称> The Mysterious Island </名称>
       <作者> Verne,Jules </作者>
       <价格> 31.00 </价格>
```

```
  </图书>
</bookstore>
```

用于输出的 bookstore. xslt 代码如下：

```
<?xml version="1.0" encoding="gb2312"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
   <html>
   <head>
       <title>练习使用 XPath 轴</title>
   </head>
   <body>
       <xsl:apply-templates select="bookstore"/>
   </body>
   </html>
</xsl:template>

<xsl:template match="bookstore">
     <xsl:apply-templates select="child::*"/>
</xsl:template>

<xsl:template match="text()">
     Textnode: <xsl:value-of select="self::text()"/><br/>
</xsl:template>

<xsl:template match="*">
     Booknode: <xsl:value-of select="self::*"/><br/>
</xsl:template>
</xsl:stylesheet>
```

在 IE 浏览器中的输出结果如图 3-15 所示。

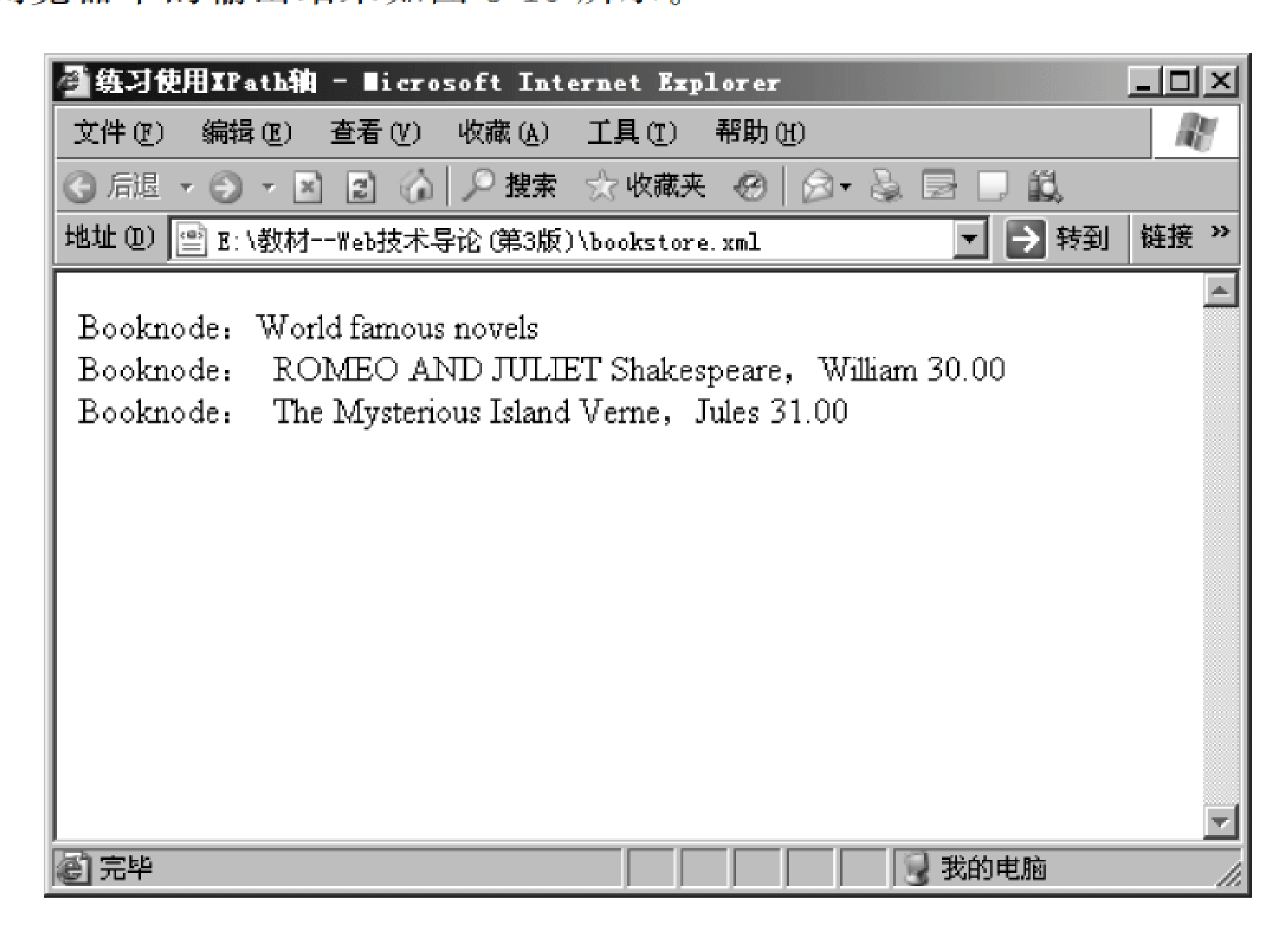


图 3-15　XPath 轴的应用输出结果界面

5. XPath 函数

XPath 定义了一组内置函数,称为核心函数库。在 XPath 1.0 中,包含 4 类函数,分别是节点集函数、字符串函数、布尔函数和数字函数。XPath 2.0 对函数进行了很大的扩充,分成 7 类,包含 100 多个函数,这些函数针对字符串值、数字值、日期和时间比较、节点操作、顺序操作、布尔值等。其中最常用的节点集函数见表 3-26。

表 3-26 XPath 常用函数

函 数 名	含 义
last()	返回上下文大小,即给定上下文中的节点数
position()	返回上下文位置,即当前节点在给定上下文节点集(列表)中的位置。比如,可以用表达式 position()=last() 测试处理集合中的最后一个节点
count(node-set)	返回实参节点集中的节点数
id(object)	返回一个节点集,根据在 DTD 中声明为 ID 类型的唯一标识符选择元素
nodeName()	返回节点的确定名称
nodeType	返回表示被选择节点类型的数值
index	返回一个节点在同一个父节点中的索引值
value	返回一个元素值的类型强制版本

关于 XPath 的函数的详细说明和使用请参考 XPath 规范。

【例 3-30】 设有一个 XML 文档 xpathtest.xml,编写一个 xslt 文档,选择所有值为 green 或 blue 的<x>元素,并输出。

代码清单: xpathtest.xml

```
<?xml version = '1.0'?>
<?xml - stylesheet type = "text/xsl" href = "xpathtest.xsl"?>
<root>
   <x> green </x>
   <y>
      <x> blue </x>
      <x> blue </x>
   </y>
   <z>
      <x> red </x>
      <x> red </x>
   </z>
   <x> green </x>
</root>
```

xpathtest.xsl 文档代码清单:

```
<?xml version = '1.0'?>
<xsl:stylesheet version = "1.0"
             xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
<xsl:template match = "root">
   <xsl:for - each select = "x | y/x">
      <xsl:value - of select = "."/>
```

```
        <xsl:if test = "not(position() = last())">,</xsl:if>
    </xsl:for - each>
</xsl:template>
</xsl:stylesheet>
```

则在浏览器中的格式化输出为：

```
green,blue,blue,green
```

3.6.4 XML 查询语言(XQuery)

现在，越来越多的信息以 XML 格式进行存储和交换，XML 数据查询成为重要的功能需求。1999 年，W3C 成立了 XML 查询工作组，开始研究制定相关标准规范。经过一个漫长的时期，2005 年 11 月，W3C 发布了关于 XML 查询的 8 个备选推荐规范。2007 年 1 月 23 日，W3C 才将 XPath 2.0 和 XQuery 1.0 确定为推荐标准。

XQuery 1.0 是 XPath 2.0 的扩展集，除了拥有 XPath 2.0 的特点外，增加了排序、重装、构造功能，而且实现了 XPath 2.0 未能实现的数据浏览和过滤方面的性能。XQuery 的优点包括：

(1) 相比 XSLT 的查询语句，XQuery 查询语句代码更简洁。XQuery 执行查询需要的代码比 XSLT 少，所以它的执行效率也高。

(2) 当 XML 数据是类型化的，那么 XQuery 是一个强类型语言，它能够通过避免非法的类型转换以及确认类型是否可以在查询操作中使用，来提高查询语句的执行效率。

(3) XQuery 能当做弱类型语言使用，为非类型化数据提供更强的功能。

(4) XQuery 正在成为 W3C 工作组的推荐语言，同时它也将会被主流的数据库提供商所支持。

由于本书篇幅所限，关于 XQuery 的基本语法和应用请参考 W3C 的具体规范。

3.6.5 可扩展连接语言

在 XML 的规范中，并没有规定有关文件链接的问题。为了使 XML 文件也能够有类似 HTML 文件超链接的功能，W3C 制定了 XML 可扩展链接语言规范(eXtensible Linking Language，XLL)规范，其分为三个部分：XLink 语言、XPointer 语言和 XML Base。其中 XLink 是规定 XML 文件之间的链接规范(和 HTML 中的外链接相似)，XPointer 是规定 XML 文件中不同位置之间的链接规范(类似 HTML 中的书签)。通过 XLink 规范和 XPointer 规范可以定义类似 HTML 中<a>标记的超链接功能，然后通过 xsl 显示该超链接。

1. XML 文档链接 XLink 规范

XLink 所设定的链接分为简单链接(Simple Link)和扩展链接(Extended Link)。其中，Simple Link 的链接功能和 HTML 的超链接基本上一样，将单个源文档和多个目标文档相链接。Extended Link 则超出了 HTML 超链接的功能，它链接的对象可以一次设定多个，允许在多个源文档和多个目标文档之间建立链接。

下面主要介绍 XLink 中的简单链接的定义和显示方法。

在 XML 文件中定义使用 XLink 元素的时候，必须要在 DTD 中声明链接元素。下列 XML 代码声明了一个 Simple Link 类型的 XLink 元素<friendlink>。然后，通过<friendlink>元素标记 XML 文档中的超链接。XML 文档名为 myfriends.xml。

```
<?xml version = "1.0"?>
<?xml - stylesheet type = "text/xsl" href = "friendlists.xsl" ?>
<!DOCTYPE myfriends[
<!ELEMENT myfriends ANY >
<!ELEMENT friendlink ANY >
<!ATTLIST friendlink
   xmlns:xlink CDATA #FIXED "http://www.w3.org/TR/xlink"
   xlink:type (simple|extended|locator|arc) #FIXED "simple"
   xlink:href CDATA #REQUIRED
   xlink:role CDATA #IMPLIED
   xlink:title CDATA #IMPLIED
   xlink:show (new|parsed|replace) "parsed"
   xlink:actuate (user|auto) "auto">
]>
<myfriends>
   <friendlink xmlns:xlink = "http://www.w3.org/TR/xlink"
               xlink:href = "http://127.0.0.1/personlists.xml"> Jane </friendlink>
</myfriends>
```

在定义一个 XLink 元素时，首先声明 XLink 名称空间，即：xmlns:xlink CDATA #FIXED http://www.w3.org/TR/xlink，该名称空间对应 XLink 规范中定义的一组元素和属性。利用这些默认属性，来定义用户的 XLink 元素属性，主要的属性如下。

(1) type：指明链接类型是 Simple Link 还是 Extended Link。

(2) href：用来设定链接的地址，与 HTML 中<a>标记中的 href 属性一样。

(3) role：声明该链接功能，提供给应用程序读取。

(4) title：声明该链接功能，提供给用户读取，与 HTML 中<a>标记的 alt 属性相似。

(5) show：有三种取值，replace 表示用链接的内容取代当前的内容，new 表示将链接的内容在一个新的窗口打开，embed 表示将链接的内容加入到当前的内容中。

(6) actuate：设置该链接是如何被激活的。auto 表示 XML 文件被解读后，链接自动被激活。而 user 表示该链接必须被用户手动激活，也就是用户必须单击一下该链接。

下面通过 XSL 将这个 XML 文件在浏览器中显示出来，即要显示上述 XML 文档中<friendlink>元素所标记的 XML 超链接。friendlists.xsl 代码清单如下：

```
<?xml version = "1.0"?>
<xsl:stylesheet version = "1.0" xmlns:xsl = "http://www.w3.org/1999/XSL/Transform"
                          xmlns:xlink = "http://www.w3.org/TR/xlink">
<xsl:output method = "html"/>
<xsl:template match = "friendlink">
<xsl:if test = "@xlink:href">
    <a href = "{@xlink:href}"><xsl:value - of select = "."/></a>
</xsl:if>
</xsl:template>
```

```
</xsl:stylesheet>
```

在浏览器中打开 myfriends.xml 文档，输出结果如图 3-16 所示。



图 3-16　XML 中的超链接

2. XPointer 规范

XML 文档是一种结构化文件，这使得借助文件结构进行内部定位成为可能，这就是 XPointer。XPointer 支持在 XML 文件中定位元素、属性、字符串等内部结构，例如，可以定位到根元素或者当前元素的某个子元素，也可以定位到文件中的一个点或两个点之间的区域。

XPointer 基于 XSL 转换中的 XPath 语言，并在其基础上进行了扩展，包括：①可以定位节点、点和区域；②通过字符串匹配定位资源片段；③在 URI 引用中定位资源片断。

由于 XPointer 的功能是文件内部定位，因此它可以使用在需要定位的任何地方，例如在可视化的 XML 编辑器中定位元素、属性等。但人们经常利用 XPointer 描述 XLink 链接的目标资源，因此，通常将 XPointer 和 XLink 放在一起讨论。

XPointer 引用的基本语法为：

```
<linkelementname xlink:href = "xmlDdocument#xpointer(expression)"> text </linkelementname>
```

其中，在 #xpointer(expression) 中，表达式 expression 是一个位置路径表达式，和 XPath 中的位置路径相同，用于定位计算，以确定被链接的 xml 文档中的位置。当利用 id(" somelocation")进行元素定位时，表示文件中 ID 等于指定常数的元素，可以将 #xpointer(id(name))简写成 #name。

例如，下面是记录个人信息的文档 personlists.xml，定义了多个人的信息，每个人都定义了 id 属性。

```
<?xml version = "1.0" encoding = "ISO-8859-1"?>
<personlist>
<person id = "Jane">
  <picture url = "http://127.0.0.1/jane.jpg" />
```

```
</person>
<person id="Cherry">
  <picture url="http://127.0.0.1/Cherry.jpg" />
</person>
</personlist>
```

在 myfriends.xml 中，修改<friendlink>元素，可以直接定位到一个具体的朋友。例如：

```
<friendlink xmlns:xlink="http://www.w3.org/TR/xlink"
    xlink:href="http://127.0.0.1/ personlists.xml#Jane">Jane</friendlink>
```

即可定位到 personlists.xml 文档中的 id 号为 Jane 的<person>节点。

XPointer 中的位置路径在 XPath 的基础上进行了扩展，提供了 5 种不同的在 XML 文件内定位的方法，包括绝对定位、相对定位、范围定位、属性定位和字符串定位，可将地址定位到相应的地方，功能上比 HTML 中的内链接更为强大，详细介绍略。

3.7 XML 开发环境 XMLSpy

XML 文档是一种纯文本文档，可以用“记事本”、UltraEdit 等文本编辑软件来编辑，但是非常麻烦，且无法进行有效性验证。Altova XMLSpy 是一个可视化的 XML 编辑和开发环境，专门用于设计、编辑和调试企业级的基于 XML 技术的应用，包括 XML、XML Schema、XSL/XSLT、SOAP、WSDL 和互联网服务技术，同时也是 J2EE、.NET 和数据库开发人员不可缺少的高性能的开发工具，可以大大提高应用系统的开发效率。

3.7.1 XMLSpy 简介

Altova XMLSpy 是工业标准的 XML 开发环境。XMLSpy 支持下列功能：

(1) 创建并编辑 XML 实例文档；
(2) DTD 编辑；
(3) XML Schema 开发；
(4) XSLT 开发和调试；
(5) XPath 开发；
(6) XQuery 开发和调试；
(7) WSDL 开发；
(8) SOAP 开发和调试；
(9) 数据库交互；
(10) Web 服务开发；
(11) Java/C#/C++代码生成；
(12) VS.NET 和 Eclipse 集成；
(13) 工程管理。

Altova XMLSpy 有多个不同的版本，其最新版为 Altova XMLSpy 2012，分为企业版和

专业版两种版本。专业版提供一个 XML 开发环境,包括 XML 文档编辑、架构设计、文件转换与调试、支持 XSLT、XQuery、databases、VS. NET 和 Eclipse 的集成等。企业版还具有自动代码生成、支持 Web 服务等功能。

关于 Altova XMLSpy 的详细信息请参考其官方网站:http://www. xmlspy. com/,可以下载 30 天软件试用版,本书将以 Altova XMLSpy 2012 企业中文版为例,简要介绍 XMLSpy 的功能及使用方法。

首先下载并安装 XMLSpy 2012 企业版,运行 XMLSpy,显示 XMLSpy 2012 企业版主界面,如图 3-17 所示。

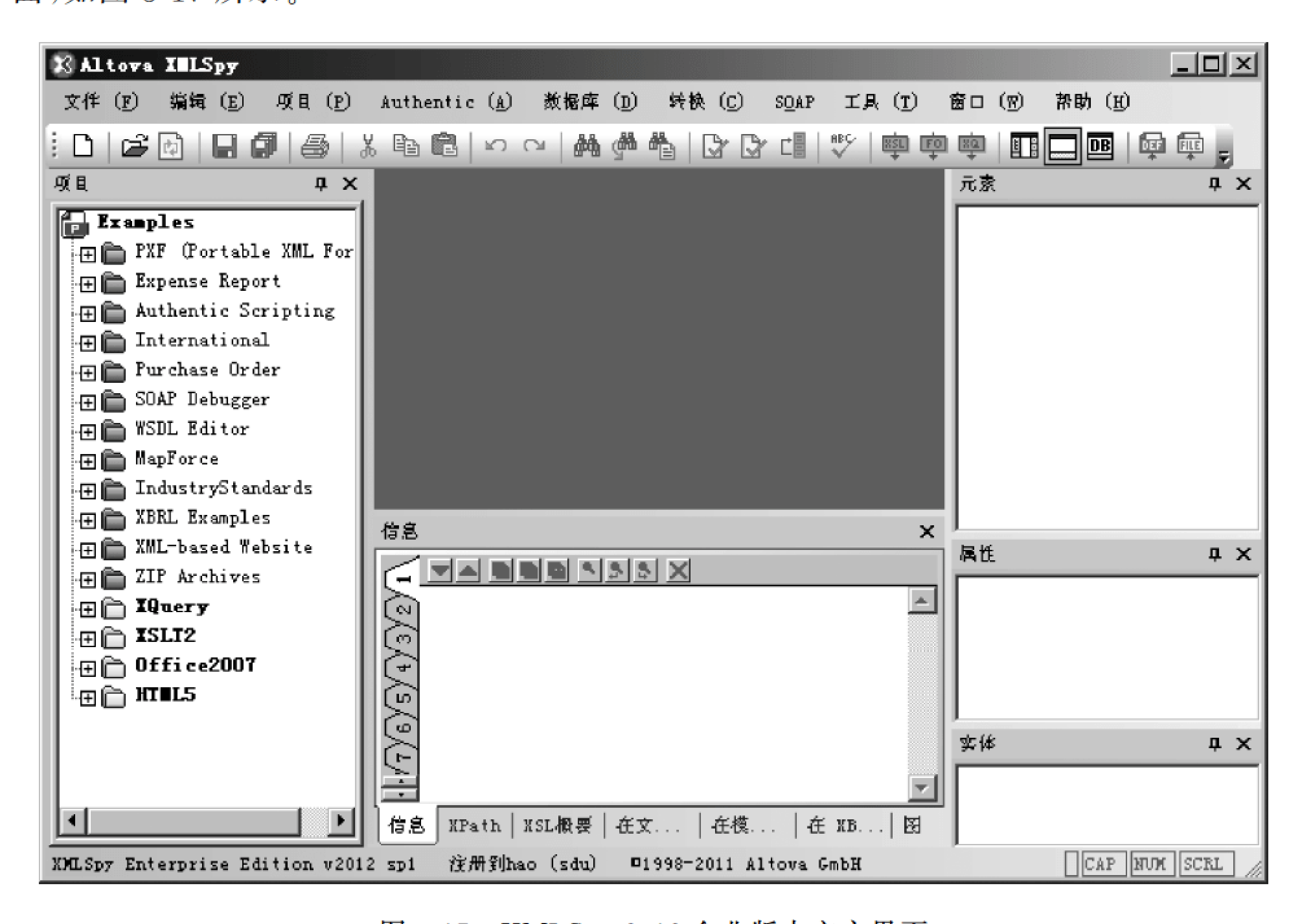


图 3-17　XMLSpy 2012 企业版中文主界面

XMLSpy 支持 XML、DTD、Schema、XSL、XSLT 等多种文件格式的编辑器。它可以将 XML 展示为完美的树形结构,可以方便地使用各种 HTML/XML/XSLT 标记,使用它可以大大节约开发时间,不必把大量的时间浪费在代码的输入上。

下面通过一个存储电影信息的实例来学习 XMLSpy 的简单使用方法。设计三个文件:films. xml、film. dtd 和 film. xslt。films. xml 负责存储具体电影内容数据,film. dtd 负责对 films. xml 进行验证,而 film. xslt 则负责对 films. xml 进行样式变换,确定它在浏览器中的最终显示效果。

3.7.2　创建 dtd 文档

下面是我们要创建的数据类型文件 films. dtd 的代码:

```
<?xml version = "1.0" encoding = "GB2312"?>
<! ELEMENT movies (movie * )>
```

```
<!ELEMENT movie (name,actor,time)>
<!ATTLIST  movie filmid CDATA #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT actor (#PCDATA)>
<!ELEMENT time (#PCDATA)>
```

要创建上述 films.dtd 文档,具体步骤如下。

1. 建立根节点 movies

(1) 在 XMLSpy 中,选择 "文件"→"新建"命令,打开"创建新文件"对话框,如图 3-18 所示。



图 3-18 XMLSpy"创建新文件"对话框

(2) 在文件类型列表中,选择 dtd Document Type Definition,单击"确定"按钮,新建一个空的 dtd 文档并在编辑区打开,如图 3-19 所示。

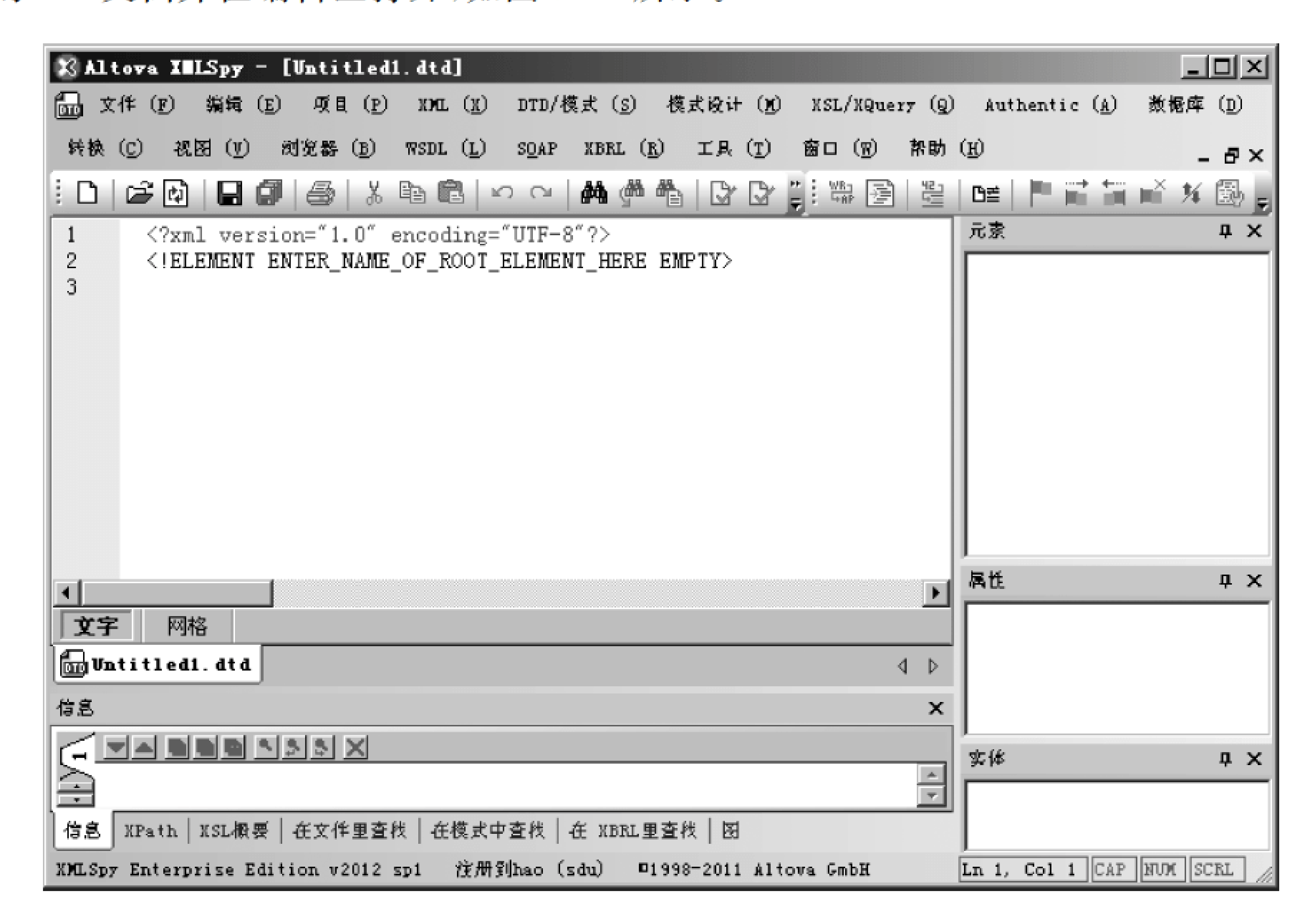


图 3-19 dtd 文档编辑界面

(3) 修改编码方式项 encoding 默认值 UTF-8，将其改为 GB2312。在编辑区，选择“网格”，切换到网格视图，如图 3-20 所示。

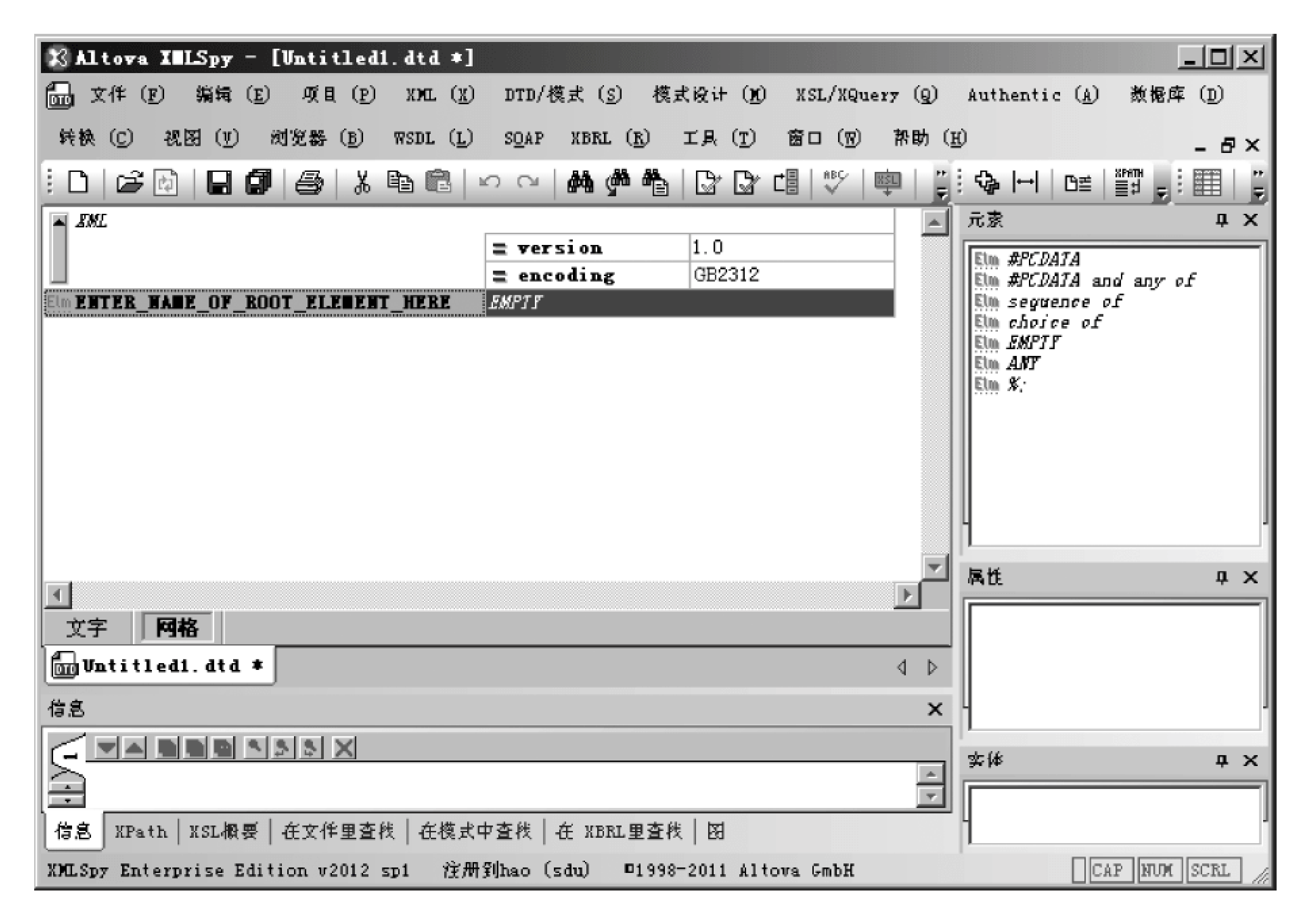


图 3-20　dtd 文档编辑界面网格视图

(4) 网格视图是一种设计视图，左侧是元素列表，右侧是元素定义，中间的分隔线可以左右拖动。当新建一个 dtd 文件后，XMLSpy 系统自动创建一个 EMPTY 元素，并给元素一个默认名，即“ENTER_NAME_OF_ROOT_ELEMENT_HERE”。在该默认名上双击，元素名进入编辑状态，输入 movies 作为元素名。这样根节点 movies 就建立完毕。

2. 定义根节点 movies 元素

(1) 根据文件 film. dtd 的设计，元素＜movies＞为一个复杂元素，由 0 个或多个＜movie＞元素组成。双击右侧“元素”窗格中的 sequence of，结果如图 3-21 所示。

(2) 接下来进行＜movies＞的定义。在 Elm 中输入子元素名＜movie＞,，然后单击元素名右侧区域，在“属性”窗格中，双击“0 or more”，如图 3-22 所示。

(3) 至此，根元素＜movies＞定义完成。单击“文字”视图，可以看到生成的代码，如下：

```
<! ELEMENT movies (movie * )>
```

3. 定义＜movie＞元素

(1) 在＜movies＞元素上右击，在快捷菜单中选择“追加”→“Ele 要素”命令，则在＜movies＞下面添加一个新的元素，如图 3-23 所示。

(2) 在新元素名称段双击，输入元素名＜movie＞，默认的元素定义为“#PCDATA”，双击，在右侧“元素”窗格，双击 sequence of，如图 3-24 所示。

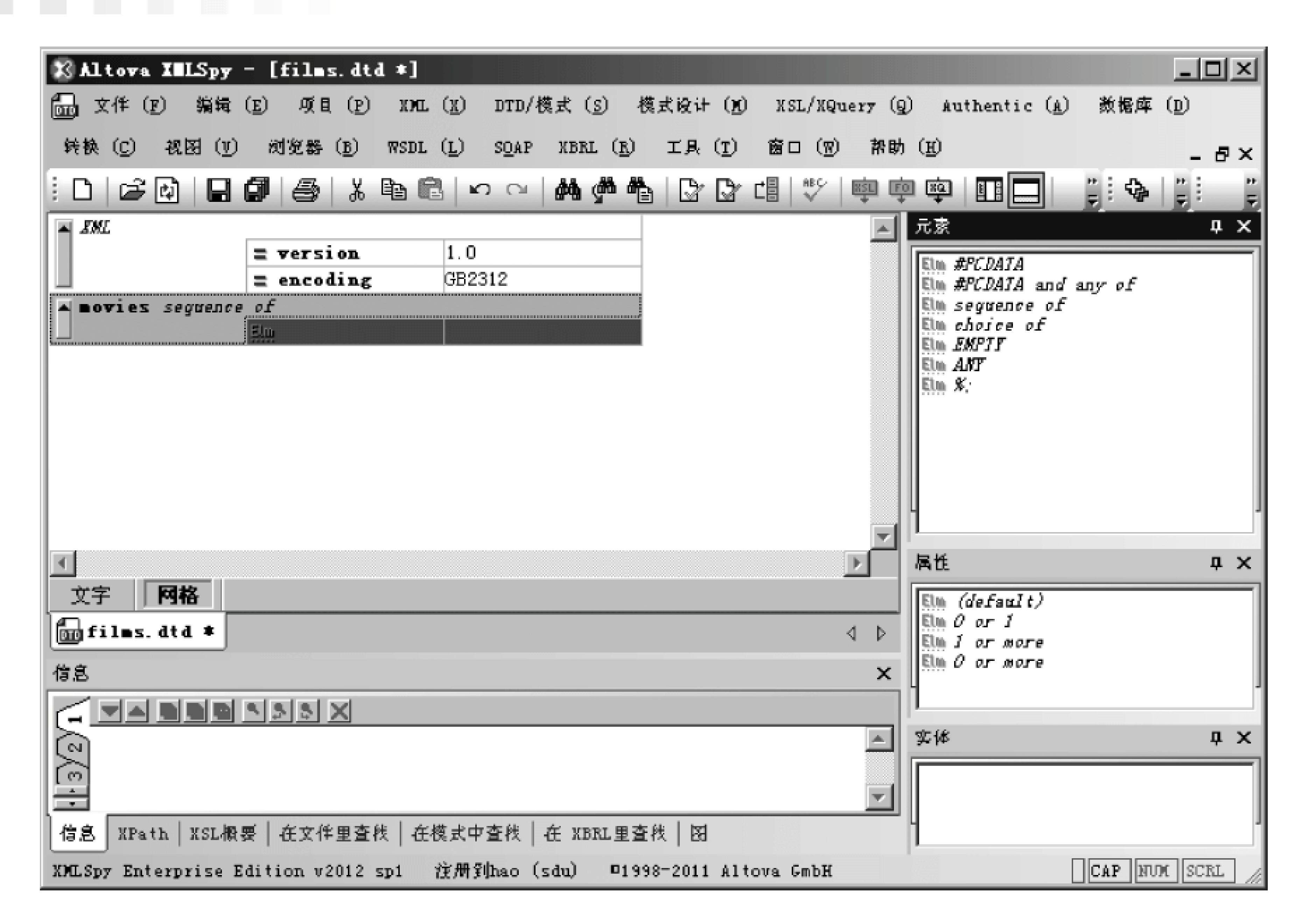


图 3-21 根节点 sequence 属性

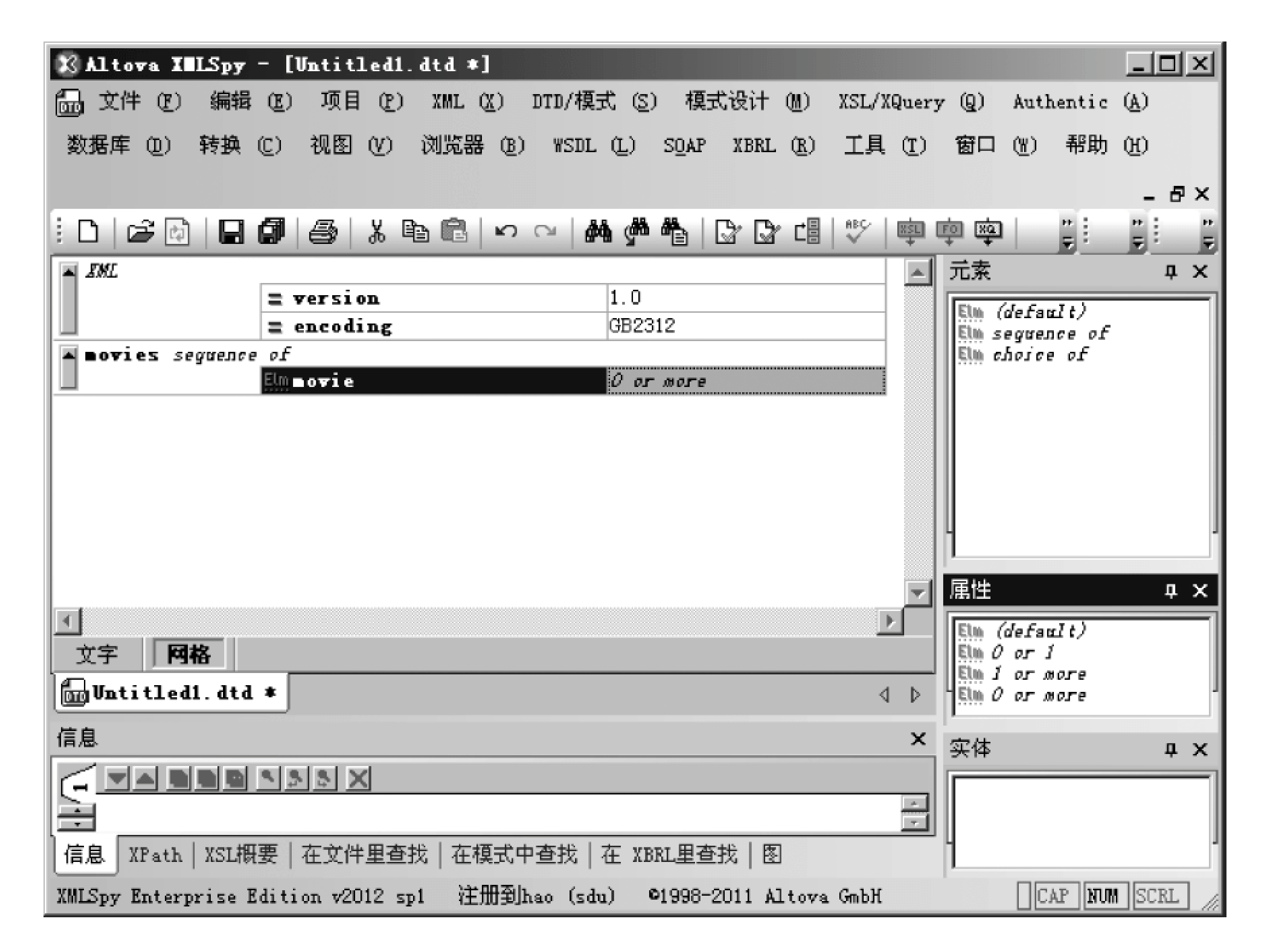


图 3-22 根节点<movies>的定义

(3) 在元素<movie>上输入第一个子元素<name>,然后执行两次“添加子元素”→“Elm 要素”命令,为<movie>元素增加两个子元素,分别输入子元素名字 actor 和 time。

(4) 为<movie>填加属性,在元素<movie>上右击,选择“追加”→“Att 属性列表”命

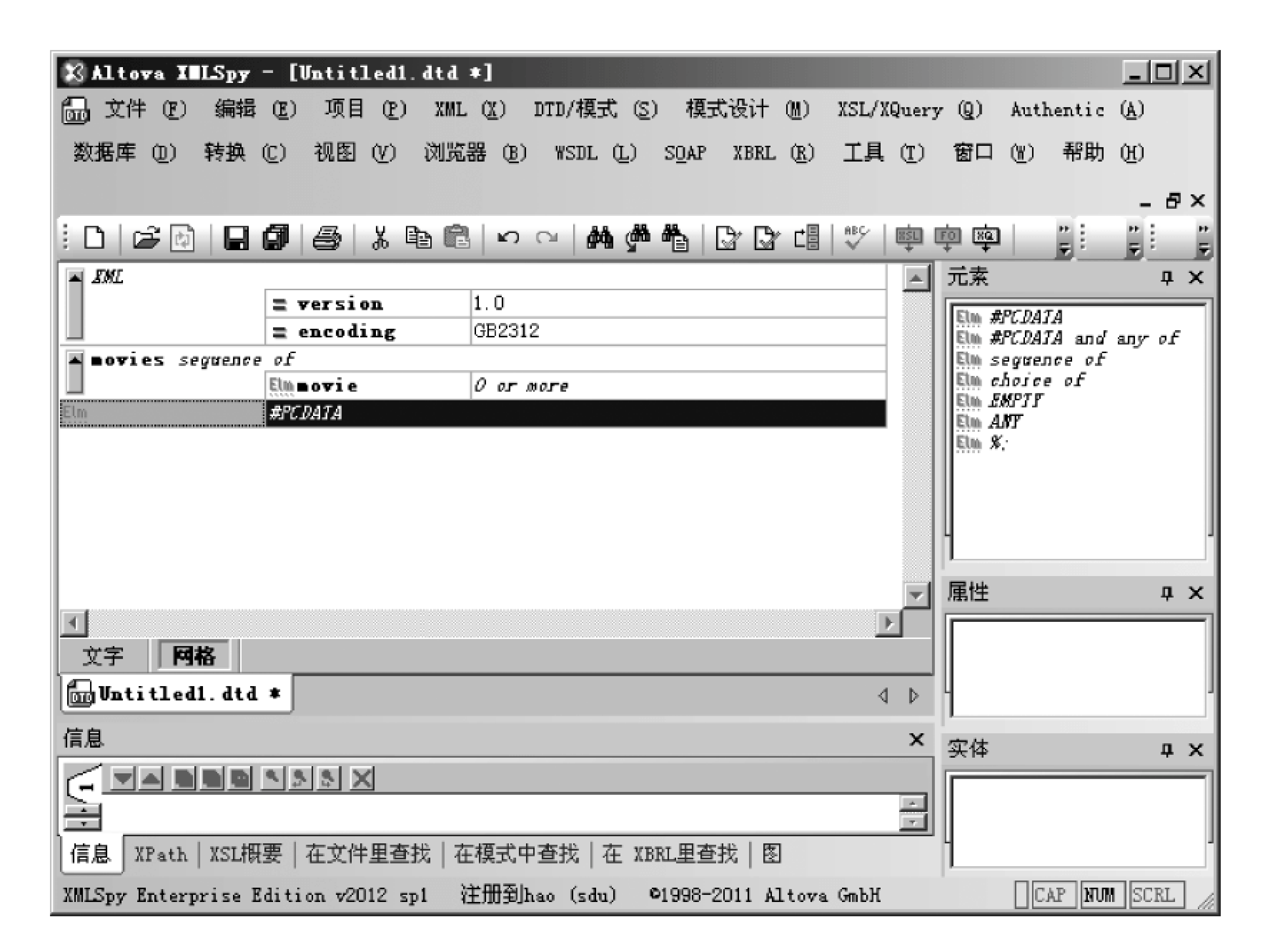


图 3-23 追加新元素

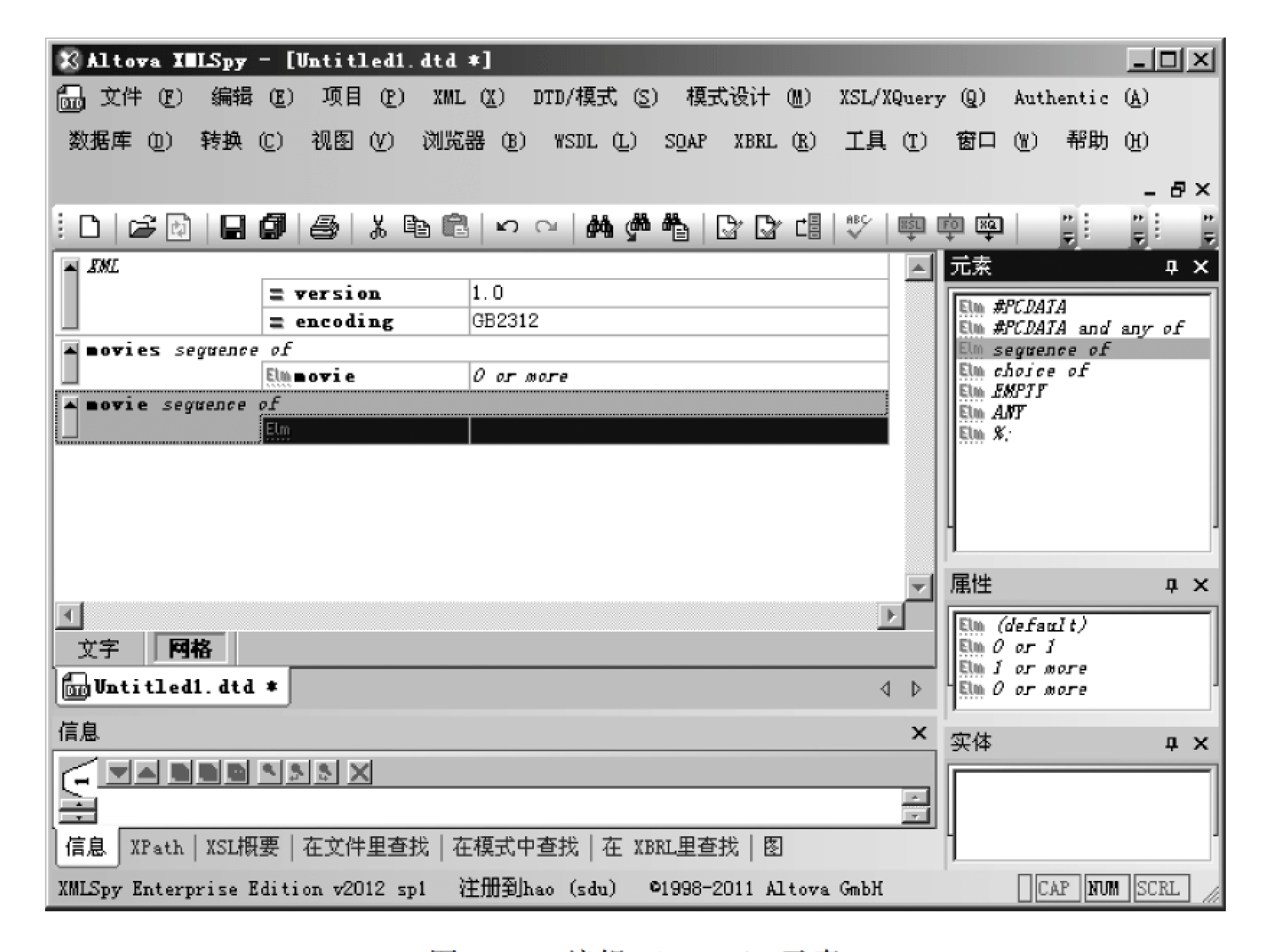


图 3-24 编辑<movie>元素

令，在<movie>元素后面，在填加元素属性编辑区，输入要为其定义属性的元素名，此时为<movie>，在列表中依次输入属性的名字、类型和取值，结果如图 3-25 所示。

（5）至此，元素<movie>定义完成了。此时，单击“文字”，切换到 dtd 编辑的文字视图，可以看到生成的 dtd 定义文本。

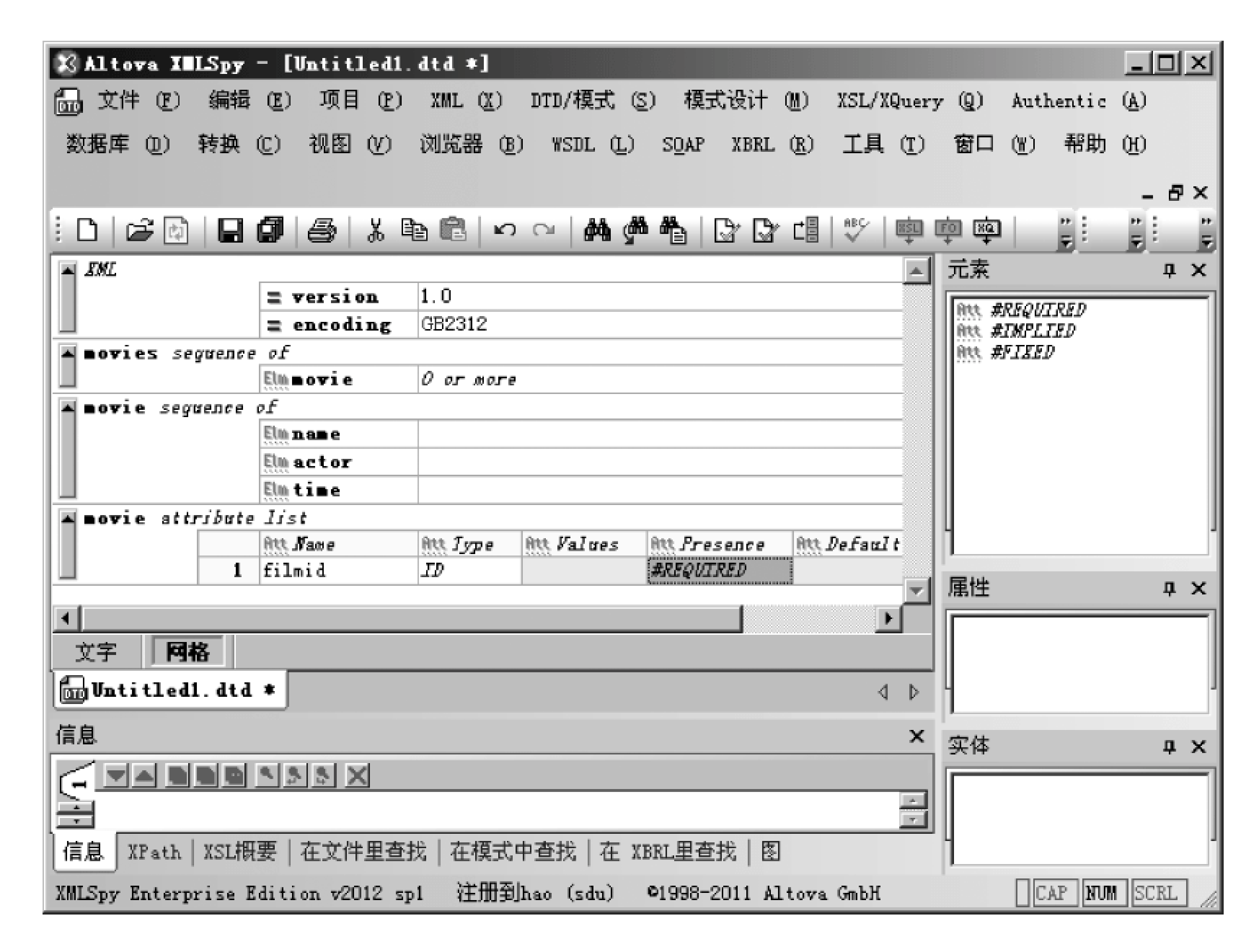


图 3-25 元素<movies>的定义

4. 定义元素<name>、<actor>、<time>

(1) 在网格视图,在元素<movie>节点上右击,在快捷菜单中选择"追加"→"Ele 要素"命令,在列表中,自动增加一个新的元素节点,如图 3-26 所示。

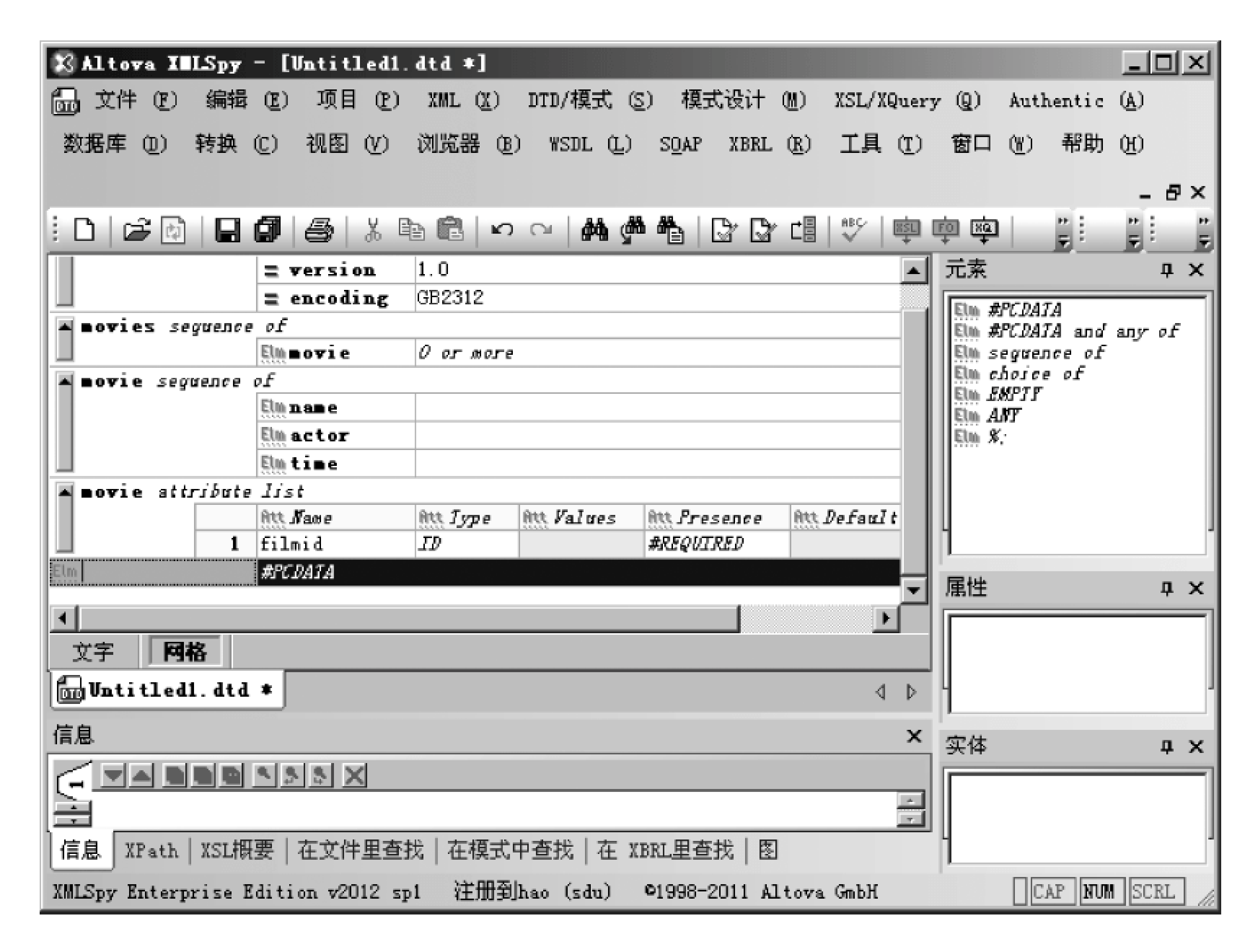


图 3-26 添加新元素

（2）在新增元素节点，单击名称区，可以输入元素名称，元素类型默认为＃PCDATA，单击该处，可以从右侧的“元素”窗格列表中，选择新的类型，如果是复杂元素，应单击 sequence of。此处，根据 films.dtd 设计，输入元素名为 name，类型为“＃PCDATA”。

（3）按照上述步骤，依次填加元素 actor 和 time，结果如图 3-27 所示。

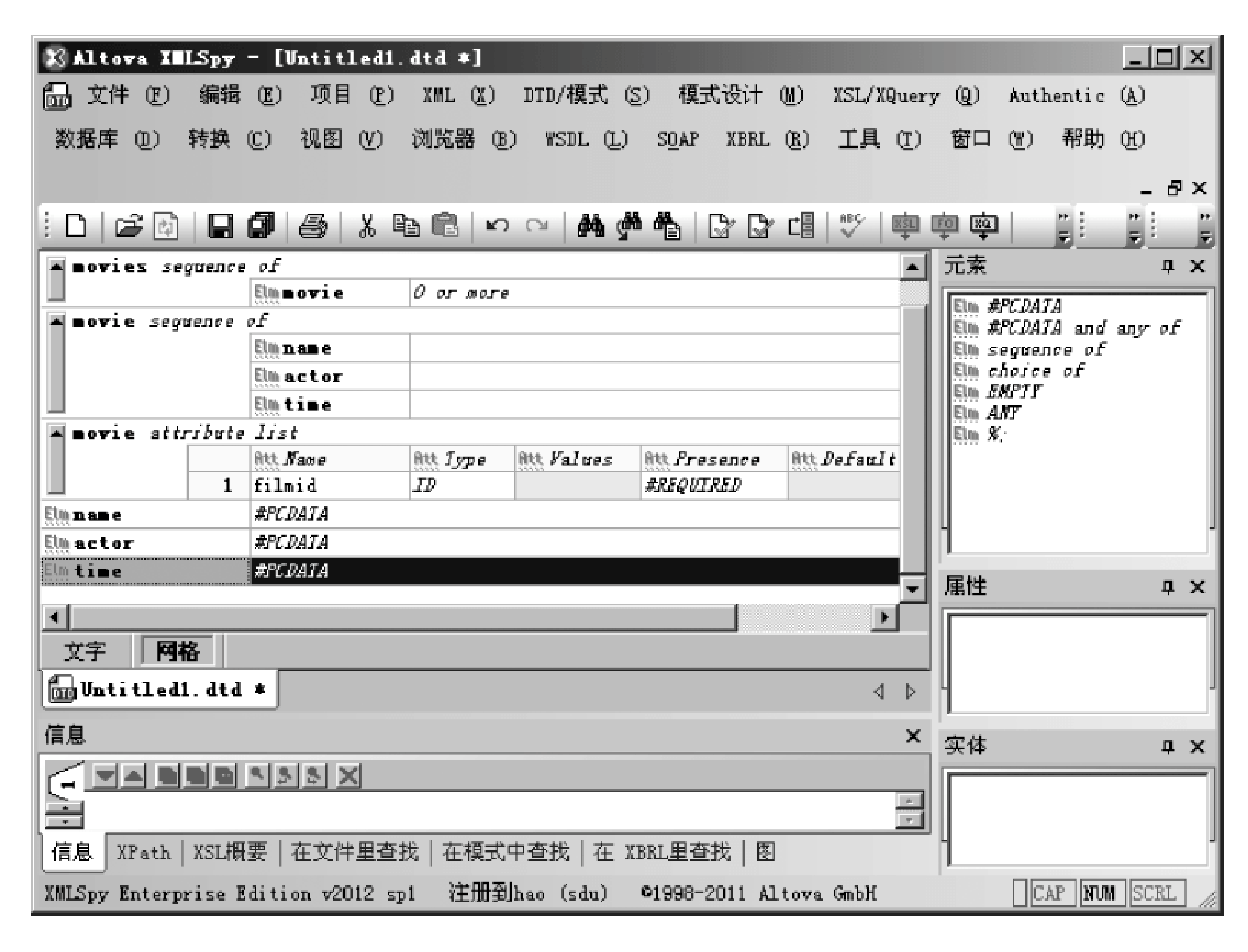


图 3-27　设计完成的 dtd 网格视图

这样 DTD 文档就建立好了。

在“文字”视图下可以查看编辑所得到的 dtd 源代码，如图 3-28 所示。

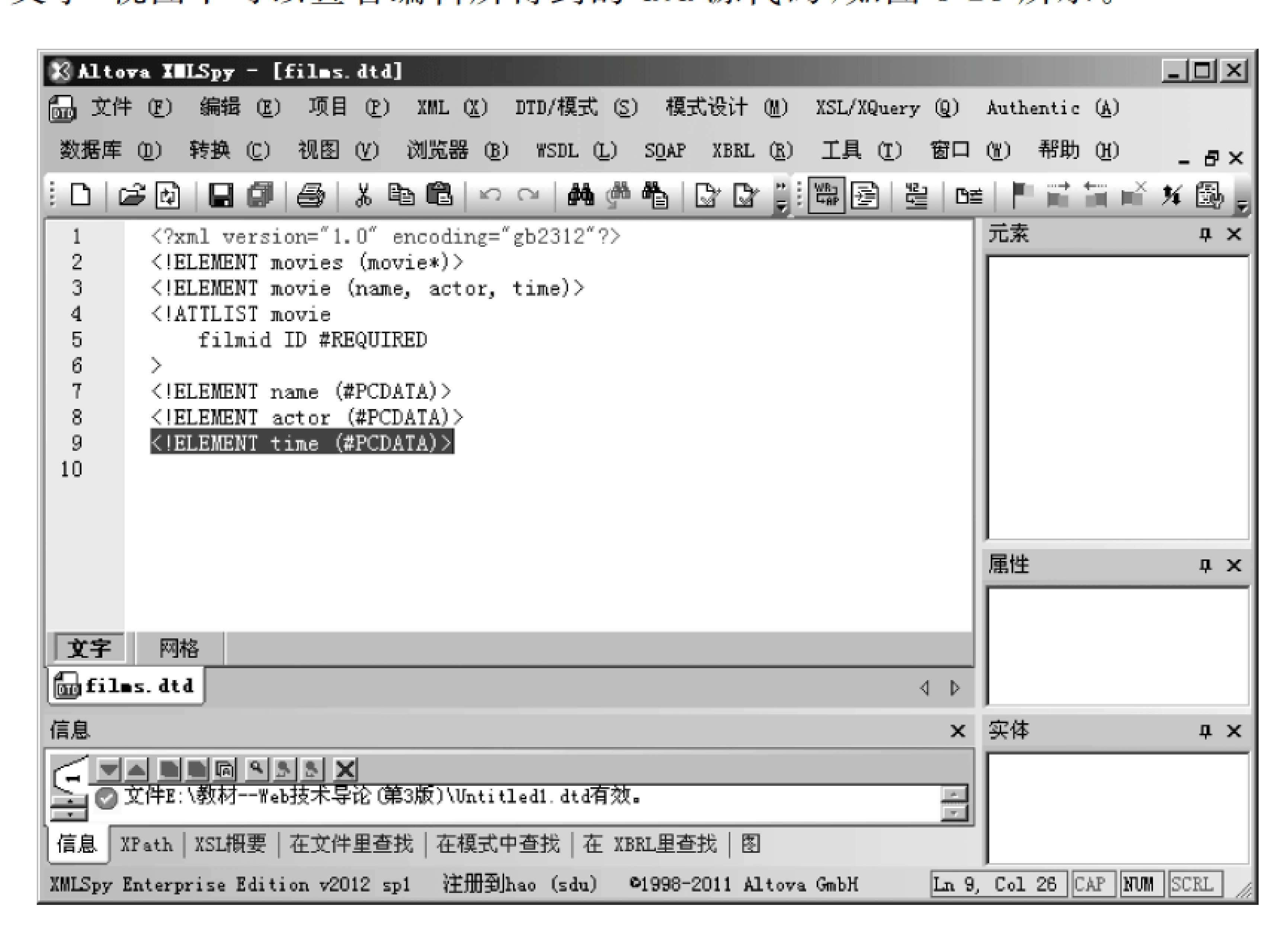


图 3-28　编辑完成后生成的 dtd 代码

(4) 最后保存文件。选择"文件"→"保存"命令,打开"保存"对话框,输入文件名films.dtd。

3.7.3 创建基于 dtd 验证的实例文档

电影文档 films.xml 代码清单如下:

```
<?xml version = "1.0" encoding = "GB2312"?>
<!DOCTYPE movies SYSTEM "films.dtd">
<?xml - stylesheet type = "text/xsl" href = "film.xslt"?>
< movies >
< movie filmid = "file - gl001">
< name >红高粱</name >
< actor >巩俐</actor >
< time > 1987 </time >
< movie filmid = "file - gl002">
< name >大红灯笼高高挂</name >
< actor >巩俐</actor >
< time > 1992 </time >
</movies >
```

具体操作步骤如下。

(1) 选择"文件"→"新建"命令,打开"创建新文件"对话框(图 3-18),选择里面的 xml (XML Document),此时会弹出一个对话框,要求选择 XML 文档的验证方式是 DTD 还是 Schema,如图 3-29 所示。

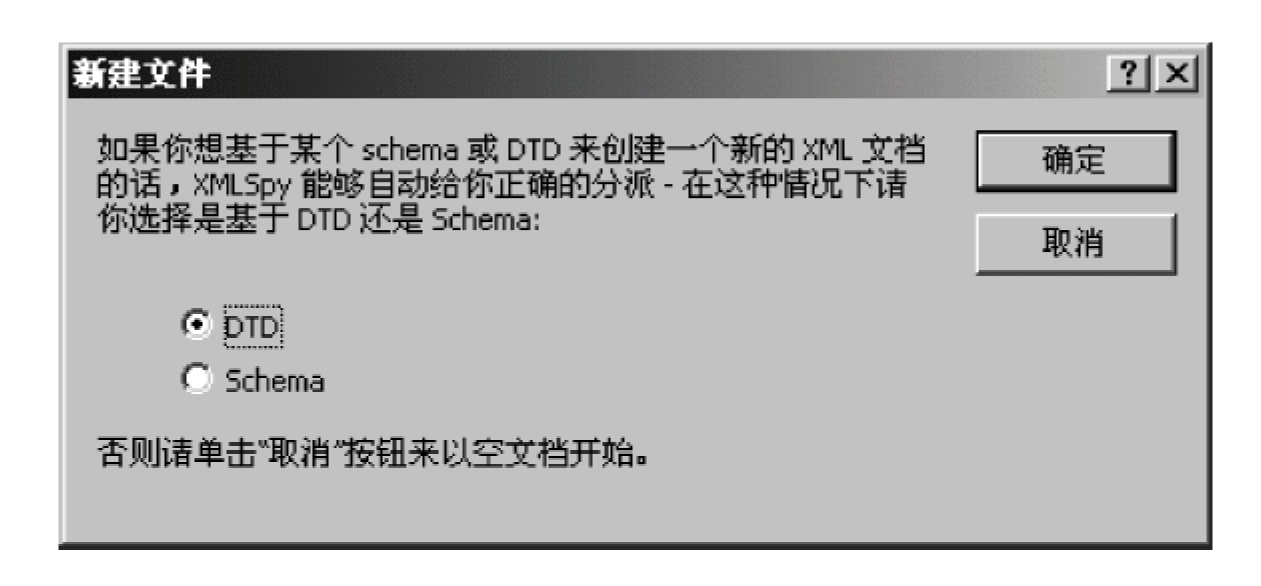


图 3-29 选择验证类型

(2) 选择 DTD 验证方式,单击"确定"按钮。然后选择刚刚创建的 films.dtd 作为其验证文档,单击"确定"按钮。XMLSpy 将自动建立一个符合 films.dtd 验证的 XML 空白文档,如图 3-30 所示。

(3) 将编码方式 encoding 更改为 GB 2312,然后输入 XML 文档内容。如果内容是已经编辑过的,可以直接复制,然后在工具栏中单击"美编"按钮,形成规范化的锯齿结构。

(4) 如果要一步步地创建 XML 文档,切换到"网格"视图,即设计视图,如图 3-31 所示。

(5) 单击根元素<movies>,在右侧的"元素"窗格,根据 films.dtd 的定义,<movies>有若干个<movie>子元素,因此,选择"添加子元素"选项卡,显示"()movie",双击该子元素,则在<movies>元素节点下,添加一个子元素节点,如图 3-32 所示。

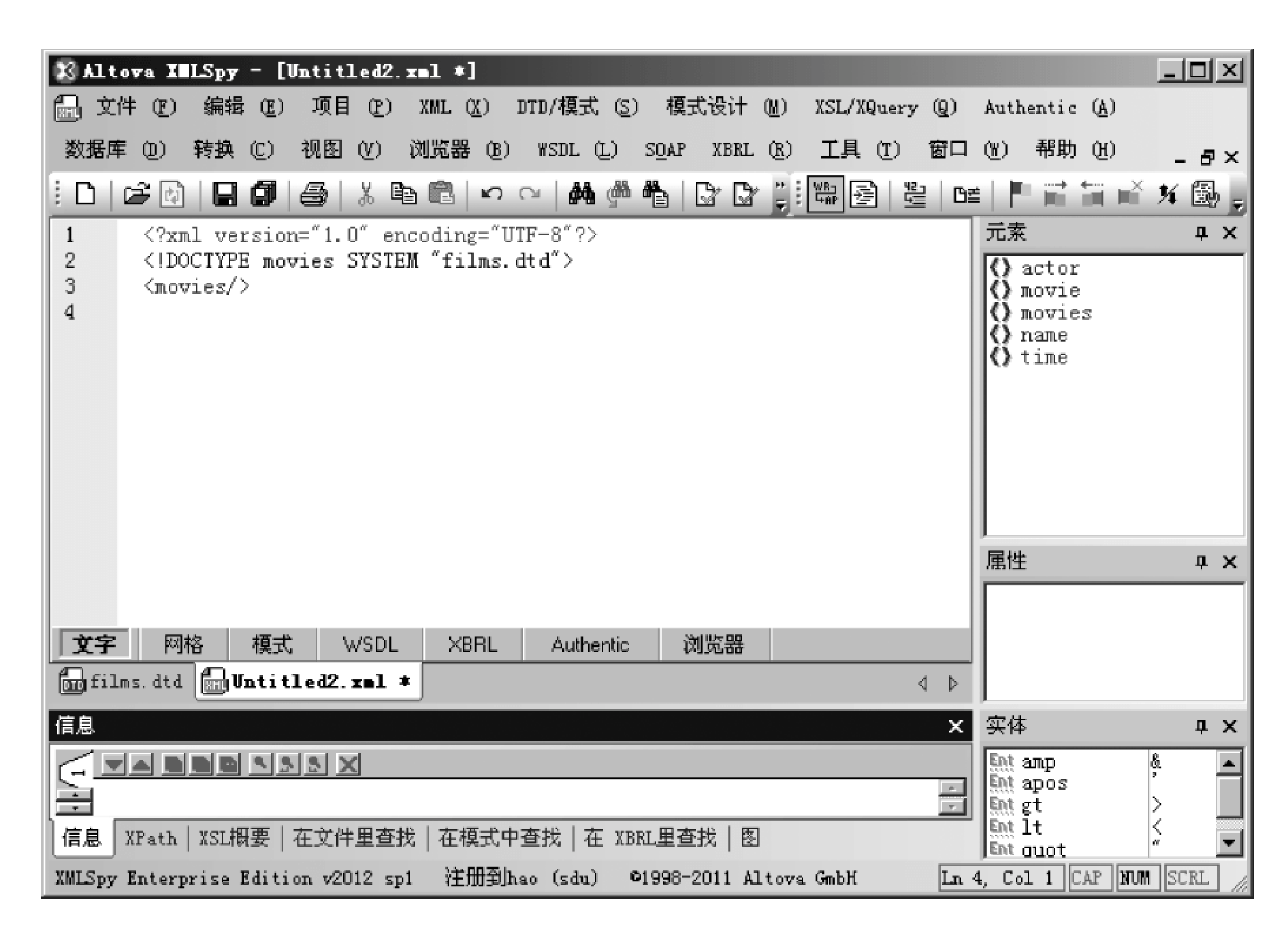


图 3-30　新建 XML 文档

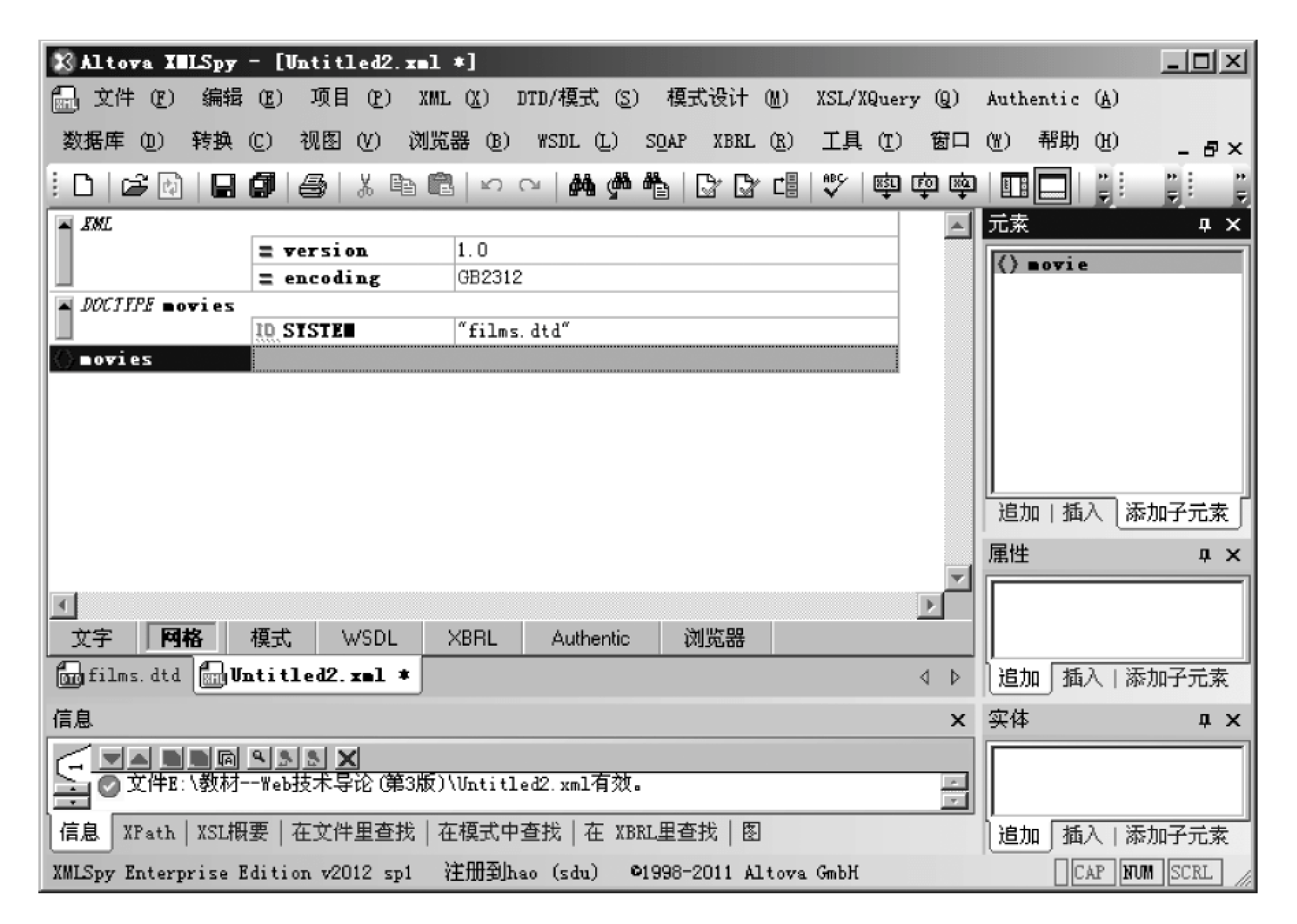


图 3-31　XML 文档网格视图

(6) 在＜movie＞元素节点，依次输入 filmid、name、actor 和 time。然后，再次单击＜movie＞元素节点，在右侧的"元素"窗格，单击"添加子元素"，再次双击"()movie"，添加第二个＜movie＞元素，再次输入元素的内容，结果如图 3-33 所示。

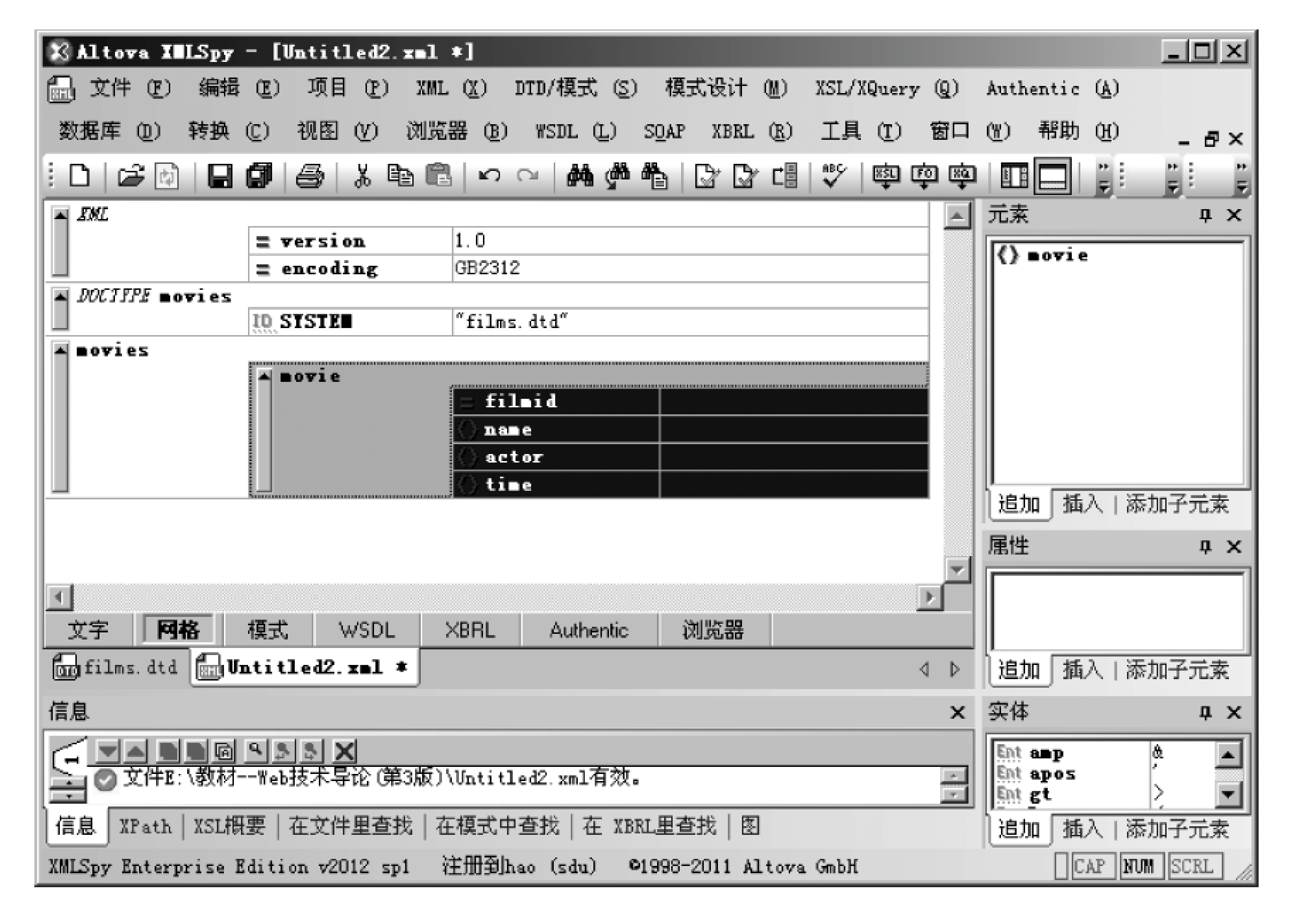


图 3-32 添加一个<movie>子元素



图 3-33 添加两个<movie>子元素的结果

(7) 切换到“文字”视图,可以看到,生成的 XML 文档如图 3-34 所示。

(8) 选择 XML→“检查良构性”→“验证 XML”命令,对输入的 XML 实例文档进行有效性验证,看其是否符合 dtd 定义。XML 文档终于编辑完毕,单击“保存”按钮,将文档保存为 films.xml。

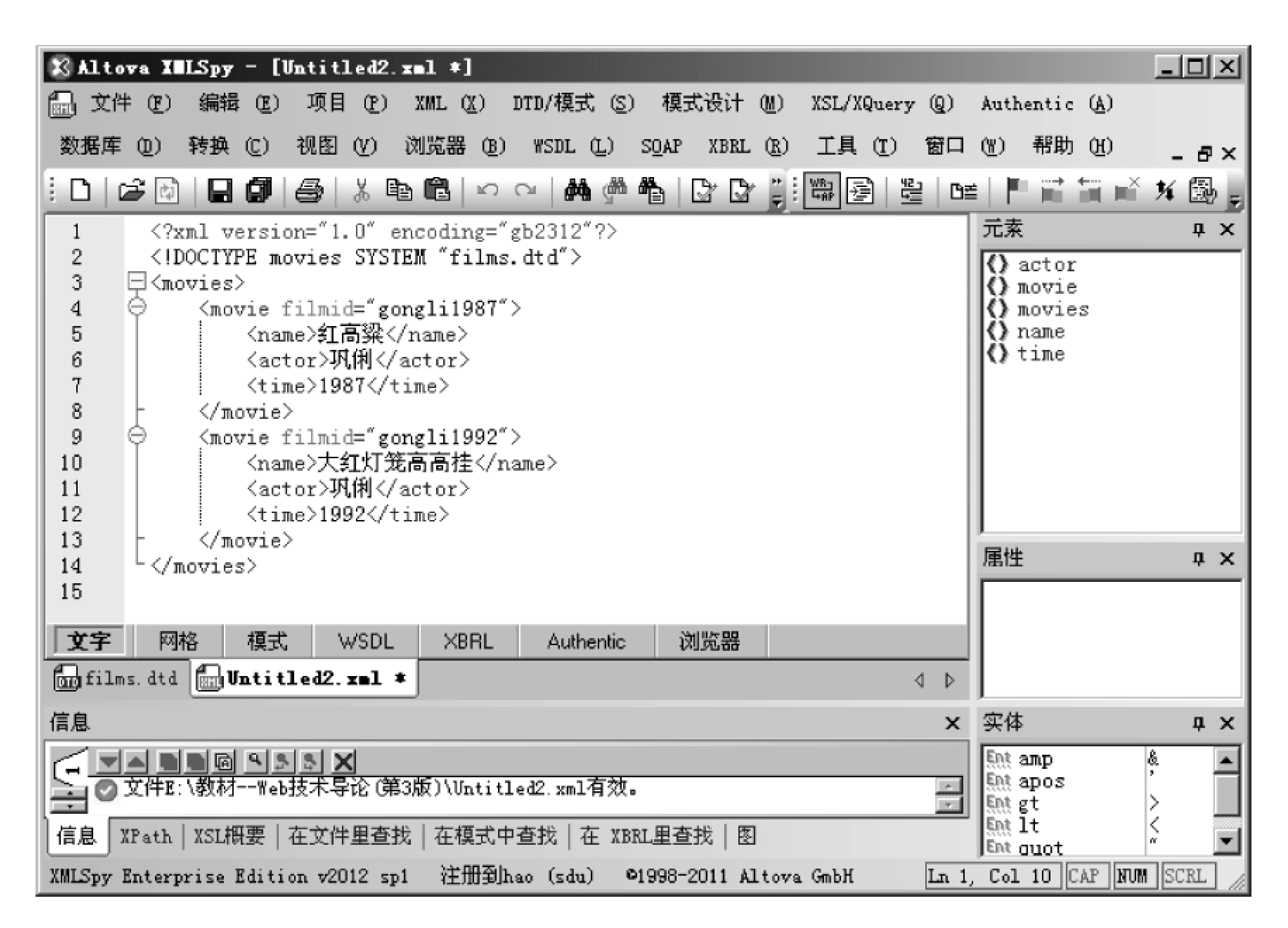


图 3-34 编辑后的 xml 实例文档

3.7.4 创建 XML 模式文档

XML Schema 可以定义元素及其类型，通过 Schema 可以构建数据模型，来约束数据存储和交换，以及实现数据的合法性校验。我们可能注意到，对于一个 XML 文档（可以看做一个 XML Schema 的实例），即使它引用了一个 Schema，当在浏览器中打开这个 XML 文档时，其存储的数据和 Schema 不一致，也不会报错，这似乎不符合 XML Schema 最初的设计思想。为什么呢？因为浏览器只检查 XML 文档的格式，并不进行数据合法性的检查。要检查 XML 文档数据的合法性，需要使用 XMLSpy。

1. 数据模型

下面通过一个例子说明 XML Schema 文档的设计和创作过程。假设对于图书信息建立下面的数据模型。

图书信息 Schema，代码清单如下（文档名 books. xsd）：

```
<?xml version = "1.0" encoding = "gb2312"?>
< xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema" elementFormDefault = "qualified"
attributeFormDefault = "unqualified">
< xs:element name = "catalog" type = "CatalogData"/>
< xs:complexType name = "CatalogData">
   < xs:sequence >
      < xs:element name = "book" type = "bookdata" minOccurs = "0" maxOccurs = "unbounded"/>
   </xs:sequence >
</xs:complexType >
< xs:complexType name = "bookdata">
```

```
        <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="author" type="xs:string"/>
            <xs:element name="publisher" type="xs:string"/>
            <xs:element name="publishdate" type="xs:date"/>
        </xs:sequence>
        <xs:attribute name="isbn" type="xs:string"/>
    </xs:complexType>
</xs:schema>
```

2. 建模过程

下面是利用XMLSpy的基本过程。

1) 新建XML Schema文件

在XMLSpy中,选择“文件”→“新建”命令,打开“创建新文档”对话框;从文件类型列表中选择xsd W3C XML Schema,创建一个新的xsd文档,如图3-35所示。

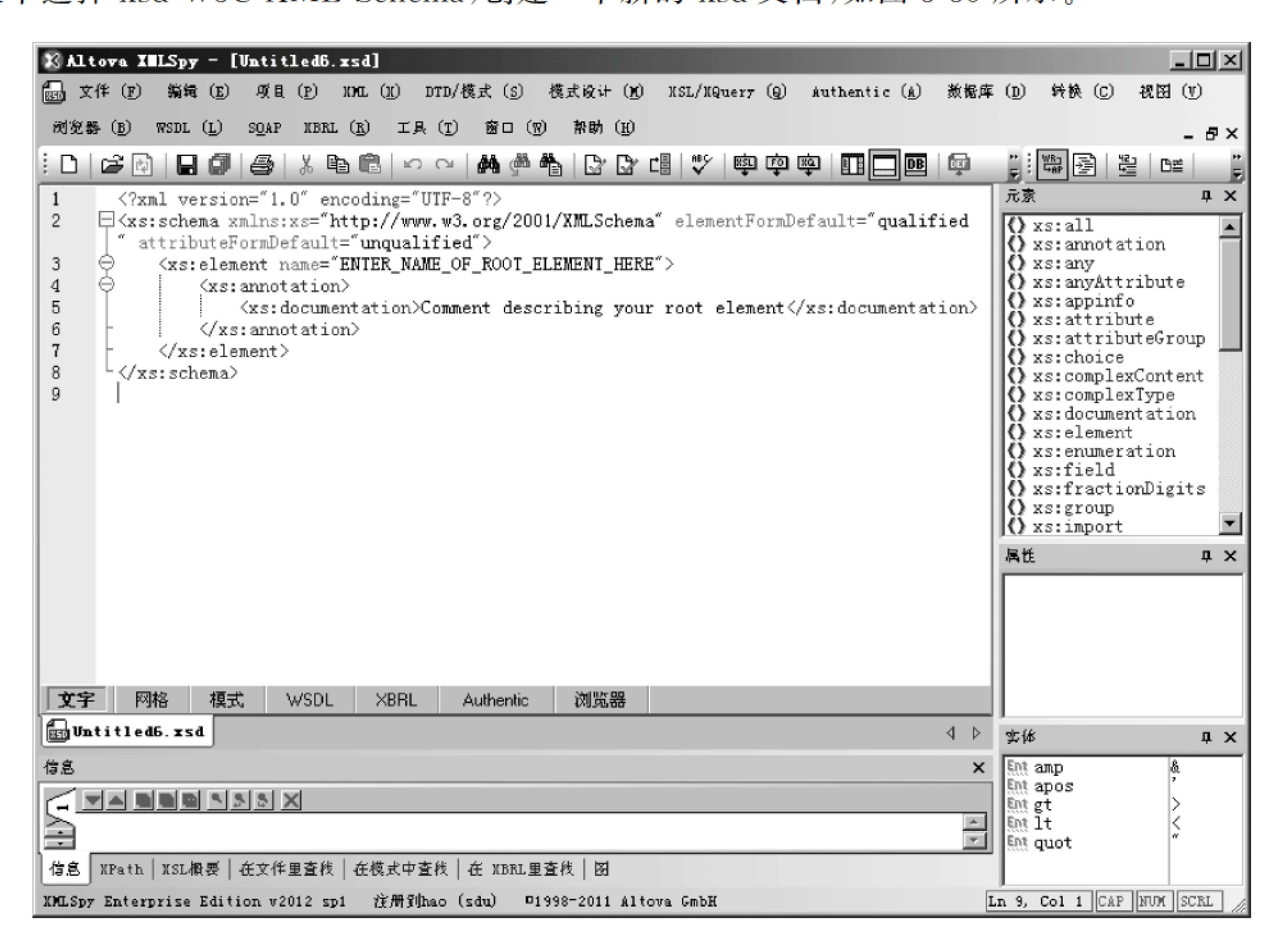


图3-35 新建XML Schema文档

新建一个W3C XML Schema文档时,XMLSpy会自动创建一个元素,名字为默认的ENTER_NAME_OF_ROOT_ELEMENT_HERE,这个元素尽量不要删除,即使你想把已经编辑好的xsd文档复制过来,此时应该复制到这个默认元素的下面,将代码清单book.xsd复制到编辑区,然后在工具栏中单击“美编”按钮。

2) xsd文件设计

如果要从头设计,单击“网格”,打开设计视图,或者按照代码清单的内容一步步地建立

books. xsd，如图 3-36 所示。

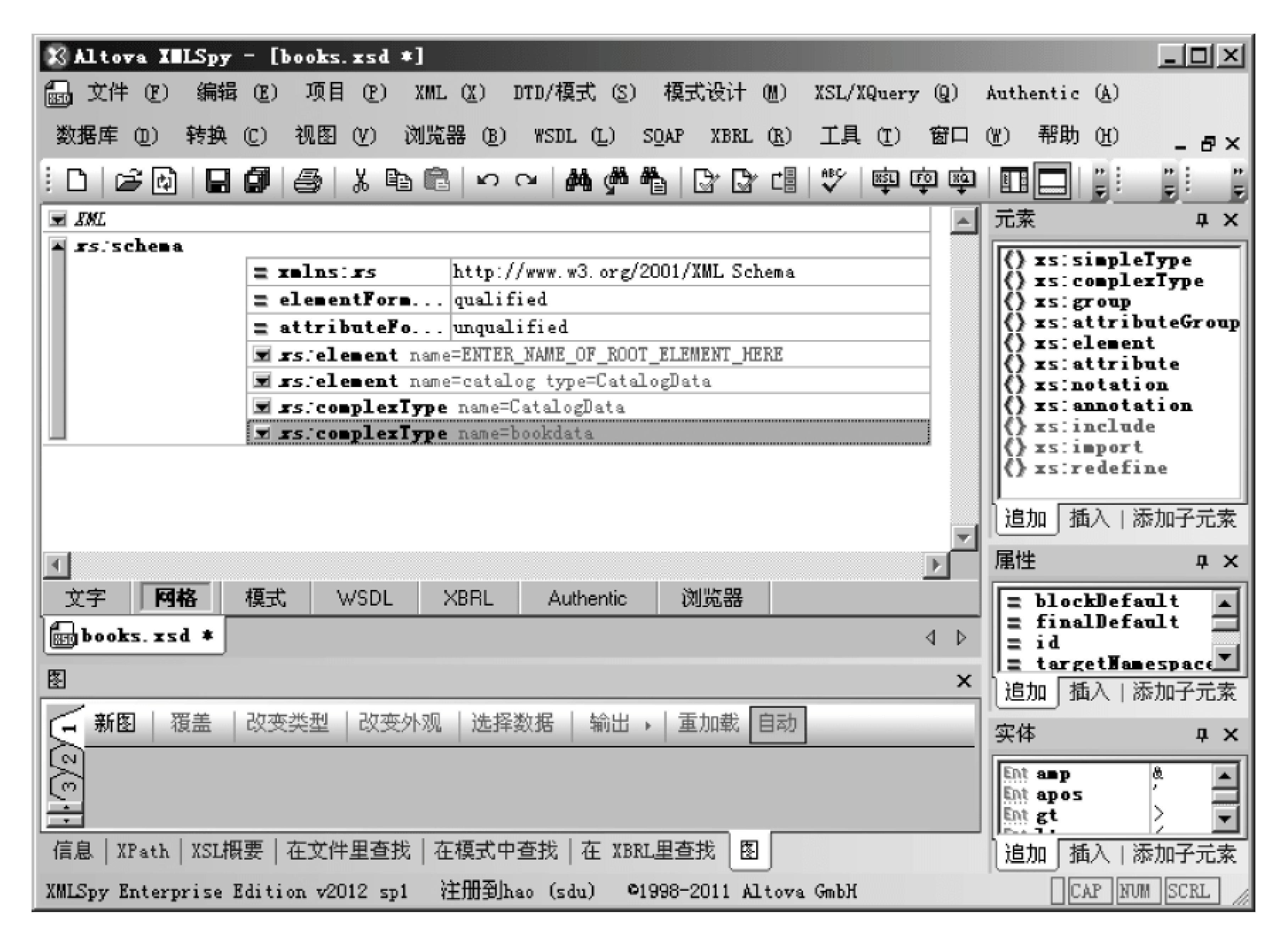


图 3-36　编辑 xsd 模式文件

在网格窗口，编辑区列出了已有的类型、元素等，此时，在右侧的“元素”窗格、“属性”窗格，列出了当前可以新建的项目，包括简单数据理性、复杂类型、组、元素等，双击即可添加新的项目。

设计完成后，可以切换到“文字”视图，查看生成的 xsd 代码，当编辑完成后，选择“文件”→“保存”命令，将 xsd 文件保存为 books. xsd。

3.7.5　新建基于模式验证的实例文档

选择“文件”→“新建”命令，打开“创建新文件”对话框；从文件类型列表中选择 xml eXtensional Markup Languagge，弹出“新建文件”对话框(图 3-18)，从中选择 Schema，单击“确定”按钮。然后，打开模式文件选择对话框，选择要应用的模式文件 books. xsd，单击“确定”按钮。

当选择模式文件(xsd)后，如果模式文件中，在＜schema＞元素内包含多个元素，则显示“选择根元素”对话框，里面列出模式文件中＜schema＞元素内声明的所有元素，此时选择我们设计的＜catalog＞作为根元素，自动新建一个 xml 实例文档框架，文档根为＜catalog＞，如图 3-37 所示。

在编辑区的下方，单击“网格”，切换到网格视图，如图 3-38 所示。

在网格视图，可以看到目前只有一个根节点＜catalog＞，它只有两个属性，没有内容。单击其中的一个属性，在右侧“元素”窗格，显示应用模式的 books. xsd 中＜catalog＞元素可包含的内容列表，根据 books. xsd 的定义，此时只有一个＜book＞元素。在“元素”窗格中，双击“()book”，则在＜catalog＞元素中增加一个＜book＞。

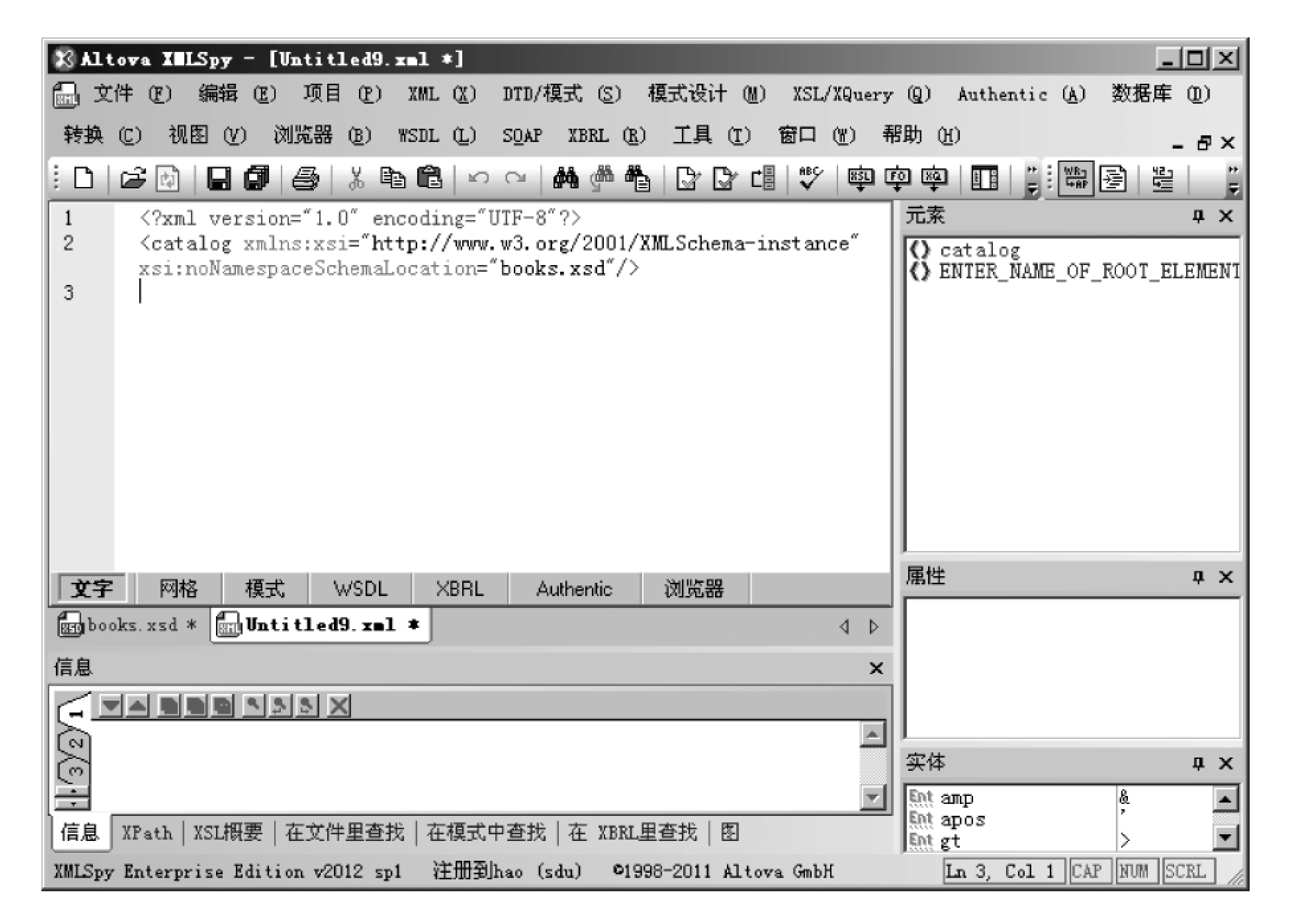


图 3-37 新建 xml 实例文档

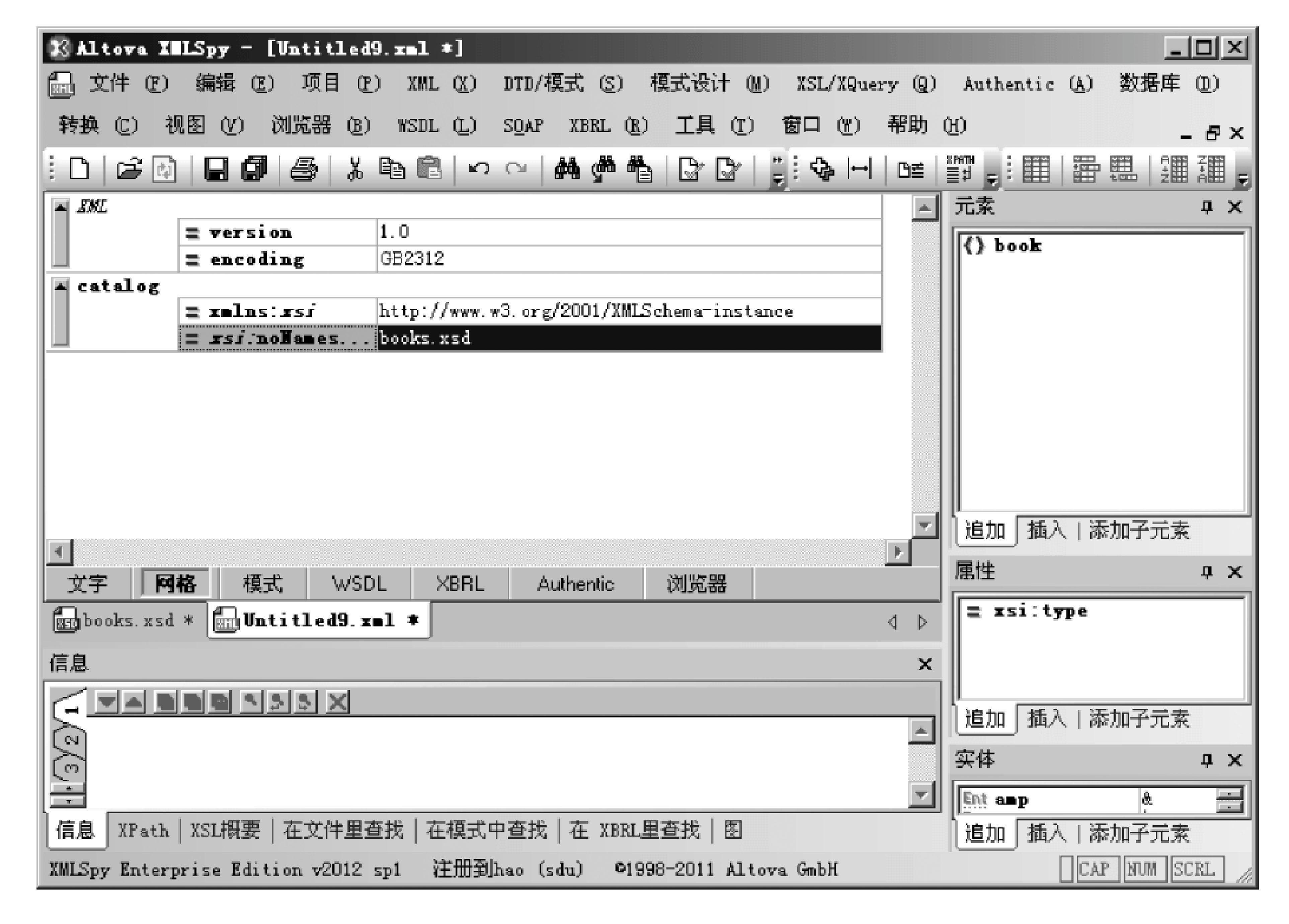


图 3-38 xml 实例文档网格视图

单击新添加的<book>元素，在“属性”窗格中，选择“添加子元素”选项卡，列出<book>元素可以添加的属性，双击 isbn，如图 3-39 所示。

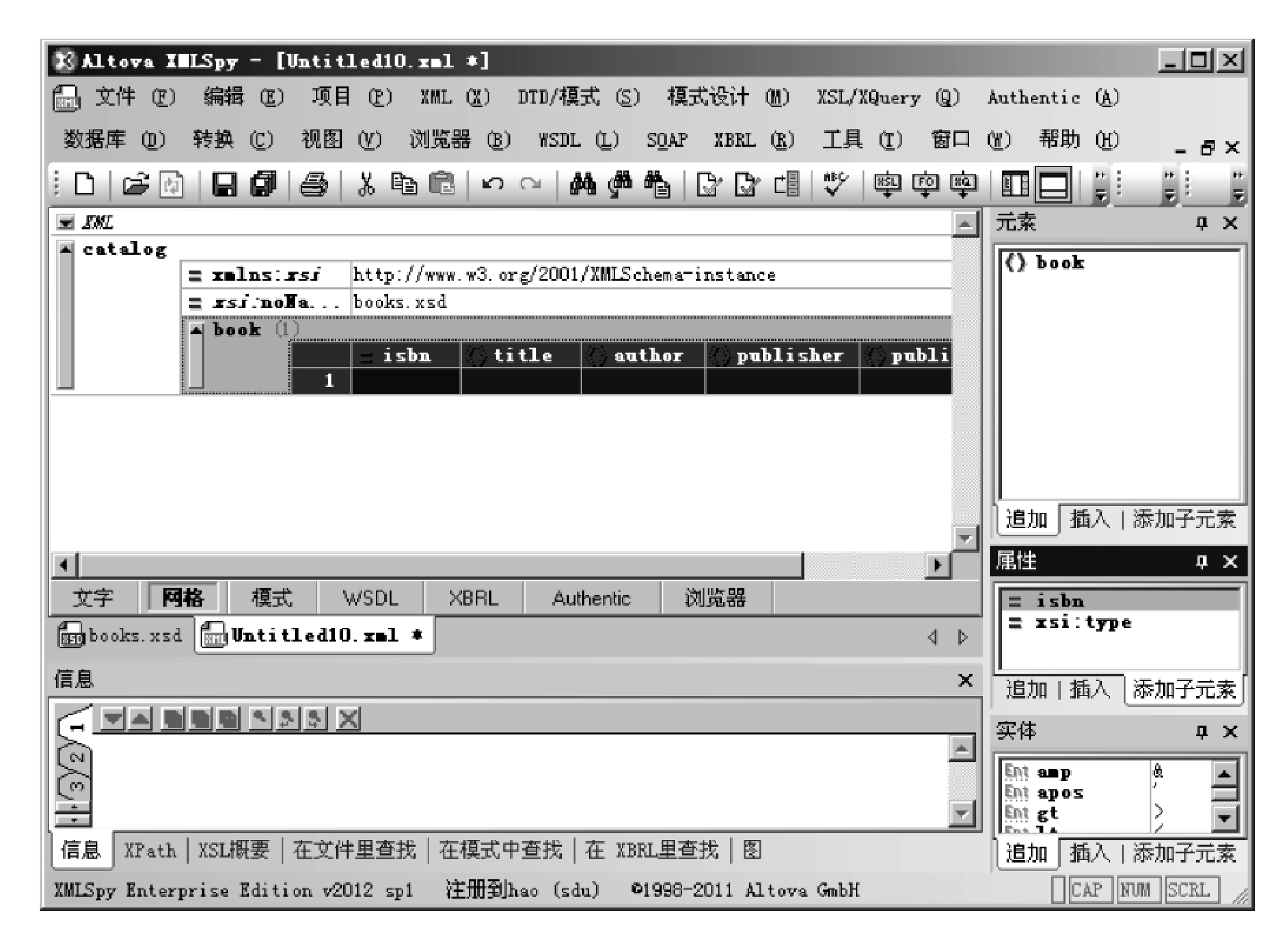


图 3-39 在根下填加元素及为元素添加属性

在<book>元素中，在每一列下输入相应的内容(属性列必须输入值)，单击“文本”，选择文本视图，可以看到生成的 XML 代码，如图 3-40 所示。

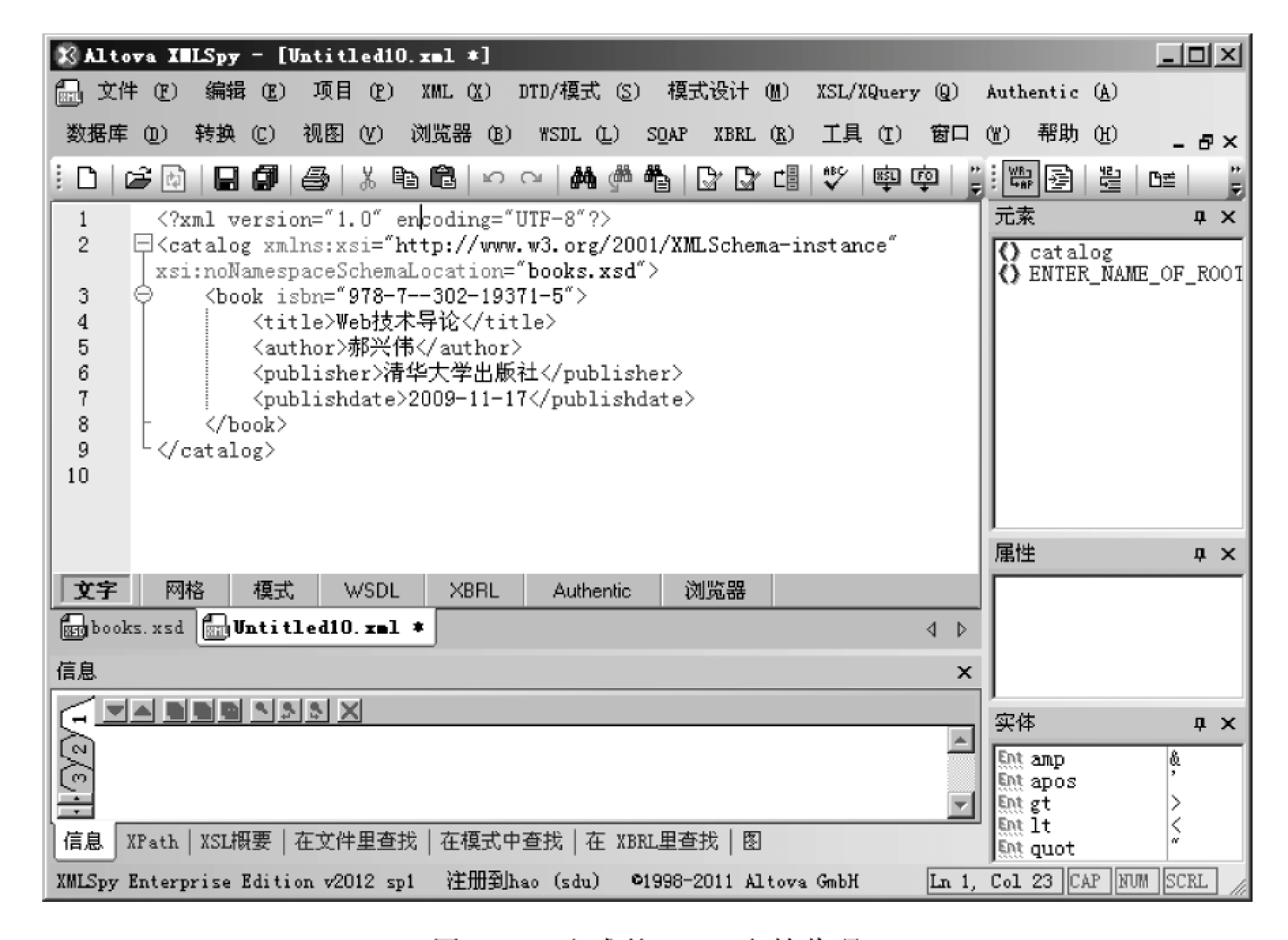


图 3-40 生成的 XML 文档代码

选择 XML“验证有效性 F8”命令,文档有效。修改日期数据为不符合日期格式的串,按 F8 键,再次验证,则系统检测到无效数据,并报错。

3.7.6 创建 XSLT 文档

当 XML 实例文档完成后,如果要进行格式化输出,还需要建立 XML 扩展样式文档,即 xsl 文档。假设要输出 films.xml 的内容,设计的 XSLT 文档代码如下:

```
<?xml version = "1.0" encoding = "gb2312"?>
<xsl:stylesheet version = "1.0" xmlns:xsl = "http://www.w3.org/1999/XSL/Transform">
<xsl:output method = "xml" version = "1.0" encoding = "GB2312" indent = "yes"/>
<xsl:template match = "/">
  <html>
  <body>
     <xsl:apply - templates select = "movies"/>
  </body>
  </html>
</xsl:template>
<xsl:template match = "movies">
  <p align = "center">巩俐代表作</p>
  <table border = "1" width = "400" bgcolor = "#CCFFCC">
  <tr>
     <td>名称</td>
     <td>时间</td>
  </tr>
  <xsl:for - each select = "movie">
  <tr>
     <td><xsl:value - of select = "name"></xsl:value - of></td>
     <td><xsl:value - of select = "time"></xsl:value - of></td>
  </tr>
  </xsl:for - each>
  </table>
</xsl:template>
</xsl:stylesheet>
```

下面是利用 XMLSpy 创建 xsl 文档的基本过程。

(1) 选择“文件”→“新建”命令,打开“创建新文件”对话框,选择里面最后一项 xslt (Extensible Stylesheet Language),打开“创建新 XSL/XSLT 文件”对话框;选择“通用 XSL/XSLT”,单击“确定”按钮,建立的新 xslt 文件,如图 3-41 所示。

(2) 编辑 xslt 文件。将其编码方式改为 GB 2312,编辑设计的 xslt 文档,或直接复制。完成后的编辑页面如图 3-42 所示。

添加完毕,设置相应的节点为各元素的属性值即可完成 XSLT 文档的编写。选择“文件”→“保存”命令,设置文件名为 films.xslt。

(3) XML 实例文档 films.xml,应用 films.xmlt 输出。

在编辑区底部,单击 films.xml 标签,切换到 films.xml 文档编辑状态,选择 XSL/XQuery→“指定 XSL”命令,在弹出窗口中选择 films.xslt 文件,单击“确定”按钮,则在 xml 文档的头部,填加如下处理指令:

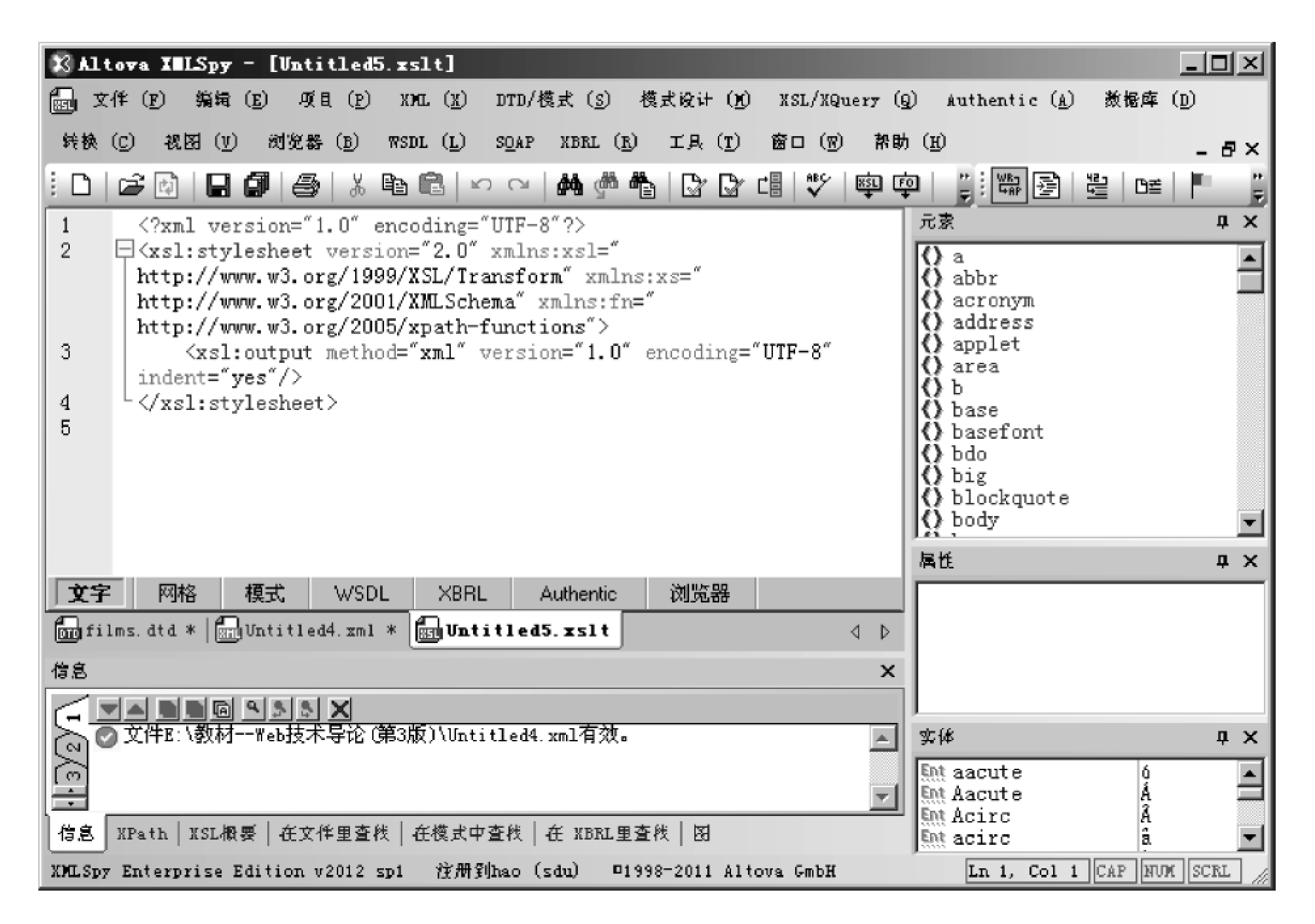


图 3-41 编辑 xslt 文档

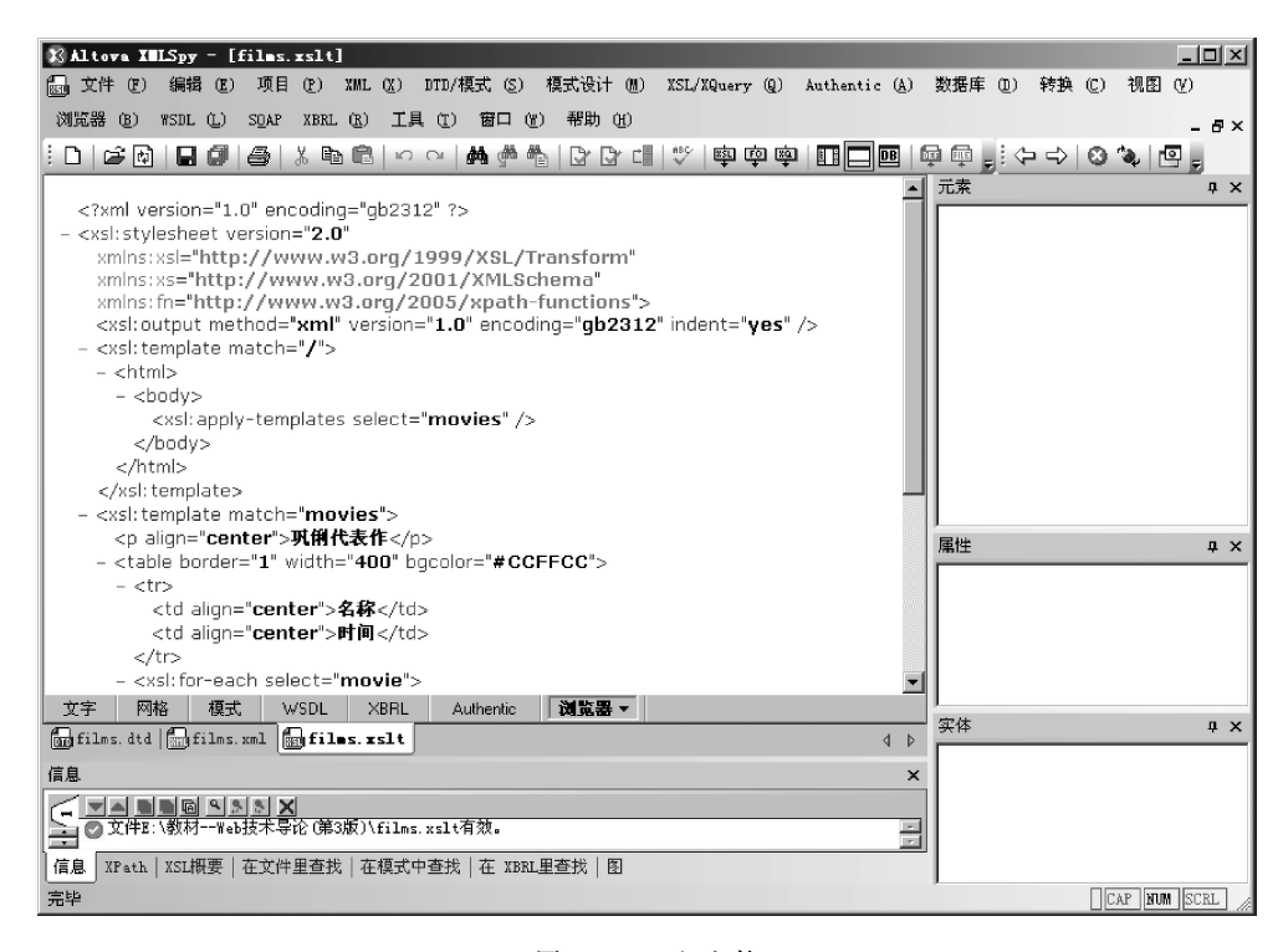


图 3-42 xslt 文件

```
<?xml-stylesheet type="text/xsl" href="films.xslt"?>
```

(4) 查看 films.xml 显示效果。在 films.xml 编辑状态,单击编辑区下方的"浏览器",即可预览当前文档的输出效果。也可以直接使用浏览器打开 films.xml 查看。如果想输出变换结果文档,在 XMLSpy 中变换后单击将结果文档存盘即可。最终显示效果如图 3-43 所示。

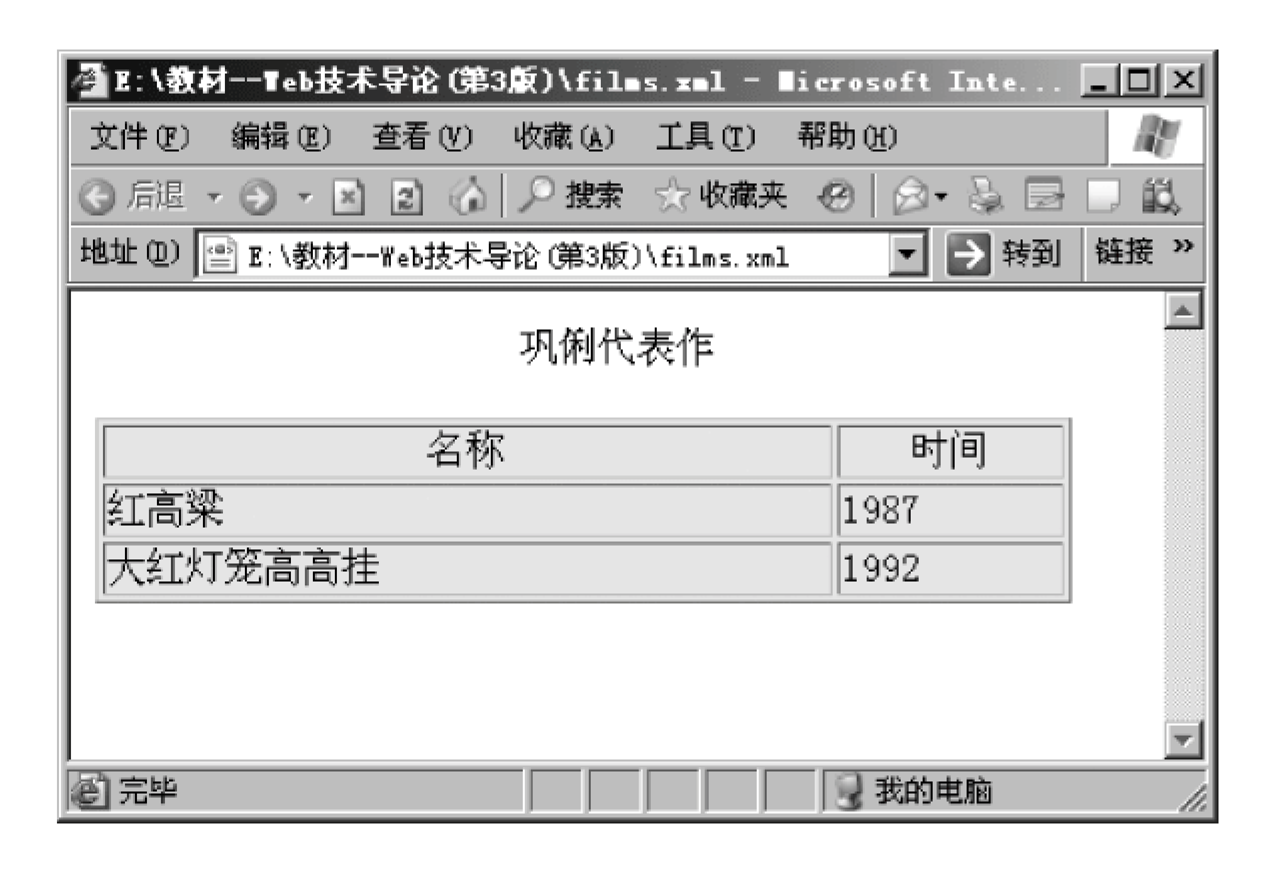


图 3-43 films.xml 实例文档在浏览器中的显示结果

本章小结

本章首先介绍了标记语言的概念、产生和发展,对几种常见的标记语言进行了简要说明,并分析了它们的定位和不同。然后重点讲解了 HTML 规范,特别是 CSS 技术和图层两大核心技术,特别讲解了改变标记默认显示样式的不同方法、各自的优缺点、块元素和行内元素的概念、图层和＜span＞的出发点和应用。对于 XML,讲解了 XML 的语法和定位,重点讲解了 XML DTD、XML 模式技术,对 W3C xsd 预定义元素和内置数据类型进行了列表总结,使得大家能够进行 XML 模式文档的定义。随后对 XSL、XPath、XQuery 等相关 XML 技术进行了简要介绍,并讲解了它们和 xml 实例文档的关系,以及它们之间的关系。最后介绍了 XML 开发工具 XMLSpy 的功能和使用,并给出了大量的案例。

习题 3

一、简答题

1. 什么是 HTML? 简述 HTML 文档的基本结构。

2. 关于 HTML 标记,回答下列问题:

(1) 标记的属性是如何分类的? 有何区别?

(2) 不同的标记,具有的属性也不相同,列举几个大部分标记都具有的一般属性,并说明这些属性的作用。

(3) 要修改标记的默认显示属性,有哪几种方法?

(4) 用 id 属性和 class 属性都可以修改标记显示样式，两者有何区别？

3. 简述 HTML 中表单<form>标记的 target 属性的作用。

4. 什么是层叠样式表(CSS)？使用 CSS 有什么好处？

5. XML 与 HTML 相比有什么本质不同？

6. 一个 XML 文档有哪几个组成部分？简述每一部分的功能。

7. 在 XML 文档中，文档类型定义(DTD)和文档架构的目的是什么？

8. XML 技术实现了数据和显示的分离，关于 XML 文档内容的显示，回答下列问题：

(1) 如果不声明 xml 文档的显示样式，在浏览器中加载一个 xml 文档时，显示的结果是什么？

(2) xml 文档树显示了文档数据的结构，如何进行 xml 文档的格式化显示？

9. 在 XML Schema，完成下列简单数据类型的定义。

(1) 声明一个带有约束的简单类型元素"age" ，age 的值不能大于 0 小于 100。

(2) 定义了一个带有约束的简单类型元素"postcode"，长度为 5 个数字字符的串。

(3) 定义了一个带有约束的简单类型"sextype"，取值为“男|女|male|female”。

(4) 定义一个复杂数据类型 nametype，包括名和姓。

(5) 扩展数据类型 nametype，新建复杂数据类型元素 fullname，包括新的元素 nickname。

10. 在 W3C XML Schema 定义语言中，<any>元素和<anyAttribute>的用途是什么？举例说明。

11. 编写关于客户资料的 Schema，用 XMLSpy 检验编写的 XML 文件的有效性。

12. 在 XML 技术中，相关的规范和标准较多，回答下列问题：

(1) 什么是 XSL？ XSL 和 XML 是一种什么样的关系。

(2) 什么是 XPath 和 XQuery？ 它们之间是什么关系？ XPath 和 XSL 有什么关系？

13. 有一个网上购物站点，针对图书类商品，包含下列信息：封面图片、书名、作者、出版社、出版日期、价格。要求：

(1) 编写一个 XML 文档，来描述图书类商品，要求对于“封面图片”定义一个图片超链接元素。

(2) 编写对应的 xsl 文件，完成图书 xml 文档的显示。

14. 对于 XML 文档，什么是结构良好(well formed)，其主要特征是什么？ 什么是文档数据的有效性？

15. 关于 Altova XMLSpy，回答下列问题。

(1) 软件有哪些典型功能？

(2) 如何验证一个 XML 文档的结构良好和数据有效性？

(3) 在工具栏中，“美编”工具按钮的功能是什么？

二、阅读理解题

1. 在 CSS 中，提供了一组定位属性，可以对块级元素和行内元素进行定位，改变正常的输出流结果，阅读下列代码，说明其输出结果。

(1) 块级元素的相对定位：

```
<html>
```

```
<head>
<style type="text/css">
p.pos_left {position:relative;left:-20px}
</style>
</head>
<body>
<p>段落的正常输出位置</p>
<p class="pos_left">段落在输出时相对于其正常位置向左移动</p>
<p>相对定位会按照元素的原始位置对该元素进行移动</p>
</body>
</html>
```

(2) 块级元素的绝对定位:

```
<html>
<head>
<style type="text/css">
h1.pos_abs{position:absolute;left:100px;top:150px}
img{
  position:absolute;
  bottom:0px
}
</style>
</head>
<body>
<h1 class="pos_abs">这是带有绝对定位的标题</h1>
<p>通过绝对定位,元素可以放置到页面上的任何位置</p>
<img class="normal" src="images/smile.gif" />
</body>
</html>
```

(3) 在文本中垂直排列图像:

```
<html>

<head>
<style type="text/css">
img.top {vertical-align:text-top}
img.bottom {vertical-align:text-bottom}
</style>
</head>
<body>
<p>图片与文字<img class="top" border="0" src="images/smile.gif" />上重齐</p>
<p>图片与文字<img class="bottom" border="0" src="images/smile.gif" />下重齐</p>
</body>
</html>
```

(4) 元素的摆放顺序:

```
<html>
<head>
<style type="text/css">
```

```
img.x{position:absolute;left:0px;top:0px;z-index:-1}
</style>
</head>
<body>
<h1>这是一个标题</h1>
<img class="x" src="images/smile.jpg" />
<p>z-index 的默认值 0,z-index: -1 拥有更低的优先级,将作为背景</p>
</body>
</html>
```

2. 在 HTML 中,经常使用 div、span 元素,并结合 CSS 的定位、浮动属性,实现特定的输出效果。阅读下列代码,说明运行结果。

(1) 图层的浮动:

```
<html>
<head>
<style type="text/css">
div{
    margin:0 0 15px 20px;
    border:1px solid black;
    width:100px;
    padding:15px;
    text-align:center;
    float:right
}
</style>
</head>
<body>
<div>
<img src="images/mydog.gif" /><br />
my dog
</div>
<p>
在该段落中,div 元素的宽度是 100 像素,它其中包含图像.div 元素浮动到右侧.同时,div 元素添加了外边距,使 div 与文本保持一个距离,此外,还向 div 添加了边框和内边距。
</p>
</body>
</html>
```

(2) 段落的首字母浮动于左侧:

```
<html>
<head>
<style type="text/css">
span
{
  width:0.7em;
  font-size:400%;
  font-family:algerian,courier;
  line-height:80%;
  float:left
```

```
}
</style>
</head>
<body>
<p><span>T</span>his is some text.在段落中,文本的第一个字母包含在一个 span 元素中.这个
span 元素的宽度是当前字体尺寸的 0.7 倍.span 元素的字体尺寸是 400%,行高是 80%.</p>
</body>
</html>
```

第4章 网页设计与制作

【本章导读】

在互联网中,网页是用户和Web进行交互的主要界面,用户通过网页来进行信息浏览和实现与Web应用的交互。网页是存储在Web服务器上的一个个html、jsp、asp、php等各种类型的文档在浏览器中的显示,一个Web应用或者一个网站是由大量的网页构成的,并通过页面间的超链接表达和实现业务逻辑。网页作为Web应用的用户界面,不仅要强调它的功能性,还必须强调它的艺术性效果,追求用户体验,这就使得Web应用的设计融入了更多非技术的要素,网页设计成为Web开发的重要内容。

本章首先介绍页面设计的概念,将页面设计分成面向业务逻辑的功能性设计(交互设计)和面向用户体验的视觉设计两个不同的层面。然后重点讨论面向用户体验的页面布局设计、页面视觉设计和页面效果设计的有关问题。接下来,对常用的网页制作工具FrontPage进行重点讲解,按照HTML规范的框架,讲解FrontPage的使用。通过FrontPage的讲解,可以使读者建立软件开发中开发环境和工具的概念,同时也使读者对HTML规范有一个更加全面的认识,特别是标记的属性,通过IntelliSense技术可以看到每一个标记的全部属性。

【知识要点】

4.1节:网页设计、页面功能与内容设计、用户体验、页面布局、视觉设计、效果设计。

4.2节:FrontPage的功能、网站。

4.3节:网页内容、插入文本、插入图片、建立超链接、插入表格、插入表单。

4.4节:标记属性、IntelliSense技术、行为面板。

4.5节:定义标记样式、用户自定义样式类、css文件。

4.6节:框架、浮动框架。

4.1 网页设计基础

和传统的程序界面不同,网页是内容和艺术的综合体。网页在实现Web应用系统功能的同时,表现出更大的灵活性、随意性和艺术性。Web开发团队的人员也更加多样,除了传统的系统设计人员、代码开发人员外,开始出现了美工人员,Web设计已经越来越明显地分成面向业务逻辑的功能性设计(交互设计)和面向用户体验的视觉设计两个不同的层面。

4.1.1 软件系统设计与 MVC 设计模式

对于所有的计算机软件系统,不论其规模和功能大小,站在概念的角度,一个计算机软件系统都可以分成两个部分,即数据和程序,所谓数据就是我们要处理的业务数据,而程序则是对数据的处理,它表达业务逻辑。从逻辑的角度讲,一个软件系统则可以分为数据、业务逻辑和连接逻辑,业务逻辑是对领域业务的编程(函数),连接逻辑是指这些业务逻辑之间的连接和调用关系(调用机制,如函数调用、消息机制等)。

在软件系统的开发中,系统设计人员追求的目标就是要保证系统的可维护性、可扩展性、灵活性等。要实现这样的目标,高内聚、低耦合是一种重要的保证。围绕这样的设计目标,出现了许多设计模式。在 Web 开发中,模型-视图-控制器(Model-View-Controller, MVC)设计模式是一种比较流行的设计模式。MVC 设计模式并不是一种新的模式,它是 Xerox PARC 在 20 世纪 80 年代为编程语言 Smalltalk-80[①] 发明的一种软件设计模式,其核心思想是在系统开发中强制性地使应用程序的输入、处理和输出分开,把数据、商业逻辑和界面显示进行分离,分别用模型、控制器和视图表示,它们相对独立而又能协同工作,各自处理自己的任务,逻辑关系如图 4-1 所示。

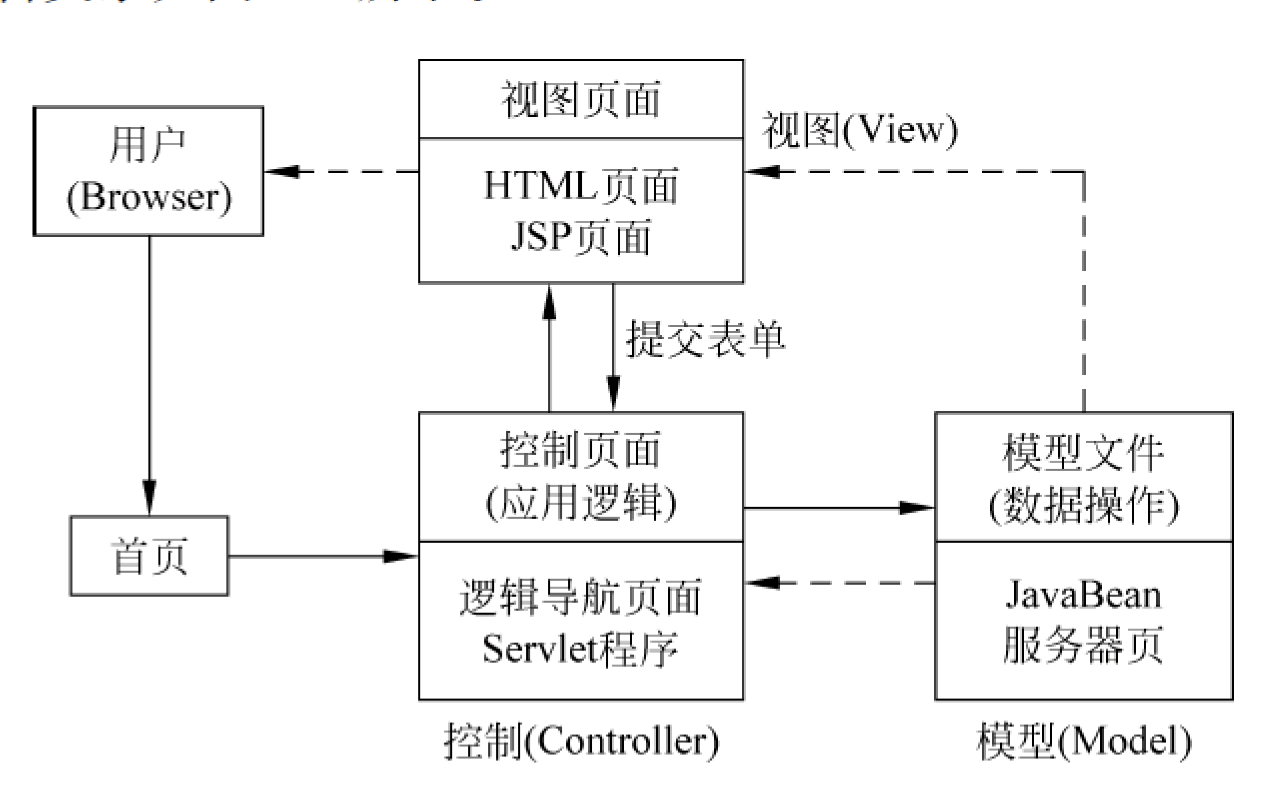


图 4-1 Web 应用中的 MVC 设计模式

模型(Model)表示企业数据及业务规则,在 MVC 的三个部件中,模型拥有最多的处理任务,实现数据操作等业务逻辑、状态管理的功能。被模型返回的数据是中立的,就是说模型与数据格式无关,这样一个模型能为多个视图提供数据。由于应用于模型的代码只需写一次就可以被多个视图重用,所以减少了代码的重复性。

视图(View)是用户看到并与之交互的界面,通常实现数据的输入(表单页面)和输出(控制模块返回的页面,模型处理结果返回页面、状态变化等)功能。

控制器(Controller)控制整个业务流程,实现 View 层跟 Model 层的协同工作。控制器接收用户的输入并调用模型和视图去完成用户的需求。当单击 Web 页面中的超链接和发送 HTML 表单时,控制器本身不输出任何东西和做任何处理,它只是接收请求并决定调用

① Smalltalk 被公认为历史上第二个面向对象的程序设计语言,是第一个真正的集成开发环境(IDE),对 20 世纪 90 年代的许多软件开发思想有重要影响。

哪个模型构件去处理请求,然后确定用哪个视图来显示模型处理返回的数据。

4.1.2 页面功能与内容设计

在 Web 应用或 Web 站点中,所有的信息和业务逻辑都通过网页来实现。页面作为用户和 Web 的界面,其功能首先是交互。因为,一个 Web 站点通常由大量的页面文件构成,因此对页面的功能划分、存储和组织应按照软件系统分析、设计和开发相关的方法和模式进行,包括生命周期法、原型法和 MVC 设计模式等。

综合利用生命周期法、原型法和 MVC 设计模式,对整个 Web 站点进行系统分析和功能设计,然后规划文件夹结构、数据库以及页面划分。和传统的软件系统开发相比,Web 站点中的一个页面,类似于传统软件系统中的一个源文件,只是页面之间的调用是通过超链接,或者页面中包含的表单的 action 属性完成的,而不是传统 C 中的函数调用。

根据 MVC 设计模式,可以将网页功能进行分类,分为:①用于输入输出的页面(视图);②服务端脚本程序页面(模型),这类页面不在浏览器中显示,主要是负责数据的查询、存储等;③导航页面,类似 MVC 中的控制器,也类似于传统程序中的菜单,实现页面之间的调用和导航,典型的导航页面就是站点首页。将网页按照 MVC 设计模式进行分类可以更好地规划一个 Web 站点,保证站点的可扩展性、灵活性和可维护性,这也是所有的软件设计模式所追求的目标。

页面的功能划分主要服务于 Web 应用的交互,提高网站的易用性,协助用户顺利地完成期望的任务流程。对于视图类和导航类页面,还必须考虑其用户体验,网页的内容设计也丰富多彩,包括网站标志、导航、菜单、图片按钮、表单样式、表格数据文字表现、新闻、公告、讨论区、Blogs、友情链接、广告条、版权信息等。对网页内容,应根据内容和用户特点,选用文本、图片、动画等不同的媒体形式来展示,以产生更好的用户体验。

4.1.3 页面布局设计

在 Web 应用中,网页是用户和 Web 的人机界面。在 Web 设计中,网页布局越来越重要,只注重内容,而忽视页面布局的网页很难产生较好的用户体验。实际情况是,用户对页面的体验第一印象就是页面的栏目和布局,然后才是页面内容。只有当网页内容和网页布局完美地融合时,才能产生最好的用户体验。

1. 网页布局设计方法

新建页面就像一张白纸,没有任何表格、框架、图层和约定俗成的东西。接下来,设计人员根据页面内容、浏览用户等因素对页面布局进行设计。进行页面布局,通常需要先在纸上或者利用 Photoshop 等画图程序来设计草稿,画出页面布局的草图,然后再深入加工,最后定稿。在进行页面布局设计时,需要考虑以下几个因素。

(1) 页面尺寸。页面尺寸和显示器大小及分辨率有关系,同时浏览器本身(如菜单栏、工具栏等)也将占去一定的屏幕空间,这在页面布局设计时必须考虑。一般情况下,以 1024×768 分辨率为标准,页面显示尺寸为 1007×600。此外,如果页面尺寸采用像素值,可以在＜body＞＜/body＞标记对内,增加＜center＞＜/center＞标记对,将＜body＞＜/body＞内

的所有页面内容居中,以应对不同的屏幕分辨率。

(2) 整体造型。整体造型是指页面的整体形象,虽然,显示器和浏览器都是矩形,但对于页面的造型,可以充分运用自然界中的各种形状以及它们的组合,例如矩形、圆形、三角形以及不规则边界的图形等。

不同的形状所代表的意义是不同的。例如:矩形代表着正式、规则,大多数政府网页都是以矩形为整体造型;圆形代表着柔和、团结、温暖、安全等,许多时尚站点喜欢以圆形为页面整体造型;三角形代表着力量、权威等,许多大型的商业站点为显示其权威性常以三角形为页面整体造型;大部分的游戏场景则使用了不规则的图形。

(3) 页头。页头又称为页眉,页头常放置站点的名字、公司标志或旗帜广告。页头的内容和设计风格直接影响到整个页面的协调性。对于站点首页,为了有效利用屏幕空间,许多网站的页头中,除了放置公司标志,往往在页头右侧还放置了一组超链接。

在页头下方,往往是一个 Flash 动画,将页头和下面的内容进行分开,从而产生较好的视觉效果。大多数的门户网站首页都采用了上述的页头设计,例如新浪、163 的首页页头。

(4) 页脚。页脚和页头相呼应,页头放置站点主题,页脚则放置制作者、公司信息以及版权信息等。

(5) 菜单。和传统的程序类似,在网页上也通常组织菜单,菜单其实都是超链接,将这些超链接组织成树状目录结构或弹出式菜单形式。

(6) 超链接。在具有导航功能的页面中(相当于 MVC 中的控制器),包含了大量的超链接,以链接到其他页面。可以按照链接的内容,将超链接组织成不同的超链接区。例如,许多门户网站的各种版块栏目,就是一个个的超链接区。超链接可以单独出现,也可以组织成多行多列的超链接区,也可以组织成横向菜单条,或者纵向的超链接区。

2. 常见的网页布局

在数以亿计的 Web 页面中,网页的布局可谓千差万别。由于网页的功能不同,表达的内容不同,人们的审美情趣不同,因此,我们不可能要求设计人员设计统一的网页布局。虽然如此,有许多的页面布局是很多设计人员喜欢使用的,如下。

(1) 基于栏目的页面布局。对于内容很多的网页,通常将页面按照栏目进行组织,组织成多个矩形区域(内容板块),每个区域包含一组超链接,形成一个超链接区。超链接区通常包含一个栏目标题,栏目之间通过色彩块来区分。为了增加视觉效果,在栏目之间,或矩形块栏目内部,通常插入一些 Flash 动画广告。

基于栏目的页面布局主要用于导航页面设计,大部分的门户网站首页即采用基于栏目的页面布局形式,例如新浪、yahoo、163、265 上网导航等。网页布局示例如图 4-2 所示。

基于栏目的页面布局主要应用于一些商业网站的首页。为吸引用户,增加用户访问量,从商业运营的目的出发,网站的内容很多,分成了不同的超链接区域(板块)。为节省屏幕空间,在栏目内往往还使用标签,以组织更多的内容。

(2) 整幅效果型布局。页面采用大幅图片或 Flash 动画,在底部加一“登录”按钮,通常用于站点首页。特点是页面美观,可以较好地展示企业形象。缺点是登录速度较慢。

(3) “口”型页面布局。页面一般上下各有一个广告条,左面是主菜单,右面放友情链接等,中间是主要内容,这种页面布局也经常用于一些站点的首页设计。



图 4-2　基于栏目的页面布局示例

“口”型页面布局的优点是充分利用版面，信息量大。缺点是页面拥挤，不够灵活。也有将四边空出，只用中间的窗口型设计。

(4)“工”型结构布局。所谓“工”型结构布局，是指将页面分成上、中、下三个部分，页面顶部为横条网站标志＋广告条；下面是版权等信息；中间为主要内容，又分为左右两个部分，左面为主菜单，右面显示内容的页面布局形式。示例如图 4-3 所示。

“工”型页面布局是网页设计中用得最广泛的一种布局方式，这种布局的优点是页面结构清晰，主次分明。缺点是规矩呆板，如果细节和色彩搭配不好，很难让人留下印象，不适宜做前卫的和个性化强的站点。

(5)自顶向下层次结构布局。该种布局形式是指将页面自顶向下分成几个平行的区域，顶部是页头，接下来的区域分别放置超链接块，最下面的区域显示具体的文章正文内容。一些文章页面或注册页面等经常采用这种类型的页面布局。

(6)自由式结构布局。上述的结构布局可称做“传统型”页面布局，还有一些页面结构布局打破了传统的页头、页尾、菜单、栏目、超链接区域等布局模式，把页面设计成一幅极具创意的广告作品。这种页面的结构布局通常用精美的图片、网站标识性图案(Logo)或变形的艺术化文字作为设计中心进行主体构图。菜单、栏目条等则按次要元素处理，自由地安排在页面上，起到点缀、修饰、均衡页面的效果。

自由式结构布局的优点是页面靓丽、现代、轻松、节奏明快，很容易让访问者驻足欣赏。缺点是下载速度缓慢，文字信息量少，信息的逻辑表达能力弱，浏览者不易直奔主题，信息查找麻烦。自由式结构布局一般用在时尚类站点上，如时装、化妆品等以崇尚现代、美感为主

题的站点,专业性的商务站点不宜采用。

页面布局设计是一项复杂的创意工作,在进行页面布局设计时,需要根据站点的性质(例如专业性商务站点、前卫现代时尚类站点等)、页面的功能、是否首页、页面内容的多少,进行精心策划,才能设计出功能性和视觉效果好的页面。

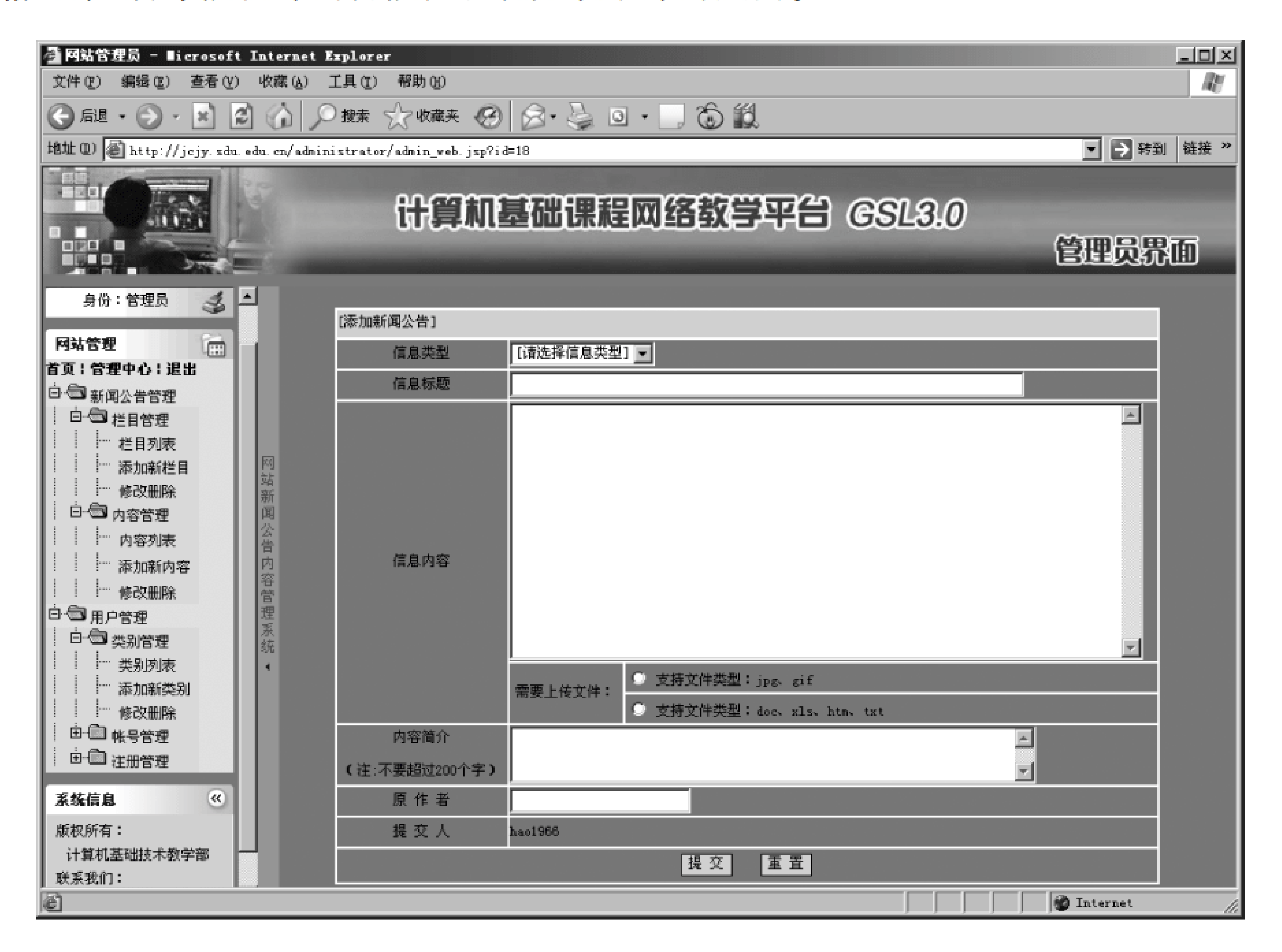


图 4-3 “工”型页面布局示例

4.1.4 页面视觉设计

视觉设计是指利用视觉符号来传递各种信息的设计,其应用的范畴很广,可以包括工业产品设计、广告设计、新媒体设计、服饰设计等。在 Web 中,则有网页的视觉设计。在网页中,视觉设计和功能性设计不同,它没有功能性要求,没有太多的理论约束,更多的是体现出感性和个性元素,服务于人的审美情趣。可以把页面视觉设计分成色彩、图形和字体几个方面。

在页面的视觉效果的诸多影响因素中,色彩设计产生的视觉效果最直接和明显,不同功能的网站,其颜色的主色调设计也不相同。大部分网站都追求一种明快的颜色设计,例如绿色、深蓝、橙色和粉红色,这些颜色都十分人气。其次是颜色的搭配,浓重的主色调表明了幽默的态度,也有助于人们迅速注意到页面上的重要元素。

关于图形的应用,现在许多 Web 2.0 站点的页面都避免使用照片,大都选择简洁的图标和截图。可以将图形的应用分为两个方面。①用于信息反馈。信息反馈一般有以下五种情况:成功、失败、询问、警告、错误/异常,视觉辅助图必须表达每种情况的准确含义。②增加趣味性。为了增加趣味性,通常用图形来表现一种状态,例如,表现喜怒哀乐的简单图形。当使用图形时,图形的设计一定要注意它的准确性,否则会起到相反的效果。例如,用一个

惊讶的表情来表示警告，但往往被误以为是询问或者出现了异常。

对于文字的字体，更多的应选择大号字。另外，要注意发挥空白的作用，巧妙利用空白可以提高页面的易读性和易用性。空白可以分离出重要信息，使眼睛得到休息，并给人以冷静和有秩序的感觉。

此外，从总体上来讲，页面设计还需要遵循几点一般性的原则，包括：①平衡，避免一边倒、头重脚轻，可以是均衡对称，也可以是不均衡对称；②重点突出，有且只有一个视觉趣味中心（Center of Visual Interest，CVI），可以通过对位置、对比和比例的处理来实现；③简化原则，去除或尽量弱化干扰，突出一点；④重复原则，通过合理重复，形成一个视觉模式；⑤统一一致，保持风格的一致性；⑥便利性、可读性与易辨认，便于浏览者阅读。

最后简单地总结一下，功能性设计通常是一种交互设计，其主要目标是提高系统的易用性；视觉设计的主要目标不是功能性的，它的目标是提高人们的视觉感受。视觉设计是一项复杂的艺术创造，需要很深厚的艺术文化积淀。虽然人们不断推崇务实的简约设计概念，但是另类的创新艺术表现也不时地出现在各种时尚和前卫的网站中。

4.1.5　页面效果设计

当网页的整体布局确定后，接下来就是效果设计了。效果设计就是利用 Photoshop 等图形图像处理工具，按照页面的布局设计，来设计页面的完整图片。然后对图片进行切图，为下面的页面 html 代码编写准备 images。图 4-4 是使用 Photoshop 设计完成的一个 Blog 页面的设计效果图示例。



图 4-4　页面设计效果图示例

当页面设计效果图完成后,接下来就是进行切图。将效果图中的元素剪切为一个个小的图片,以保证网页的浏览速度。首先要分析页面效果图,它是未来网页的显示效果。对于上述的页面效果图,切图时应从以下两个方面考虑。

(1) 在页面上部是一行带有背景的菜单,因此,可以从中切一个像素宽度为1的小图片,在页面html代码中,利用该图片实现table单元格中的背景横向填充,以达到效果图的显示效果。如果需要纵向填充,可以取高度为1的小图片,进行纵向填充。

(2) 对于页面主体中的一些栏目标题,可以直接切为小图片,这些图片的文件很小。

其他的页面元素,操作类似。最后将这些切图图片存储到一个特定的images文件夹中,它是下一步进行html页面设计的素材。

在页面效果设计中,还涉及了其中的文字,对于文字内容,可以通过CSS技术,来定义文字的样式。如果布局采用Table,表格属性要定义CSS,从而实现页面布局和显示样式的分离,增加页面维护的灵活性。关于CSS的使用,参考第3章。

4.2 使用 FrontPage

FrontPage是微软开发的一种可视化的网页制作和站点管理工具。FrontPage的功能可以分为站点管理和网页制作两个大的方面。FrontPage有FrontPage 2000和FrontPage 2003两个常用版本,两者没有本质的区别。下面以FrontPage 2003为例,重点介绍网页制作工具的网页制作方法。

4.2.1 FrontPage 主窗口

运行FrontPage 2003,FrontPage打开后,首先需要打开一个网站(即打开站点对应的主目录文件夹),一个站点不一定是一个Web服务器上真正运行的网站,可以是本地机上的一个文件夹。接下来可以在该文件夹下新建网页,或对网页进行维护。主界面如图4-5所示。

在FrontPage 2003主窗口中,可以显示网站文件夹列表,双击某个网页文件,在右边则打开该文件。在窗口的右侧是任务窗格,列出了我们最经常要做的事,比如打开网站、打开网页、新建网站网页等。这是一项非常良好的设计,它比传统的利用“文件”菜单或工具栏按钮更方便,有些软件在工具栏的最右侧会安排类似的功能。对任务窗格,可以通过“视图”→“任务窗格”命令打开或关闭。

在网页文件设计窗口的左下角,有一组网页视图模式的选项标签,分别是“设计”、“拆分”、“代码”和“预览”四种显示模式。如果是框架网页,还含有其他的几个标签。单击标签,可以选择不同的工作模式。

1. 设计模式

设计模式是一种可视化的网页设计模式。在设计模式下,用户可以采用“所见即所得”的方式设计、编辑和修改网页,系统将自动生成对应的HTML代码。代码可在代码模式或拆分模式下显示。这是FrontPage工具的核心功能,即从设计到代码生成。

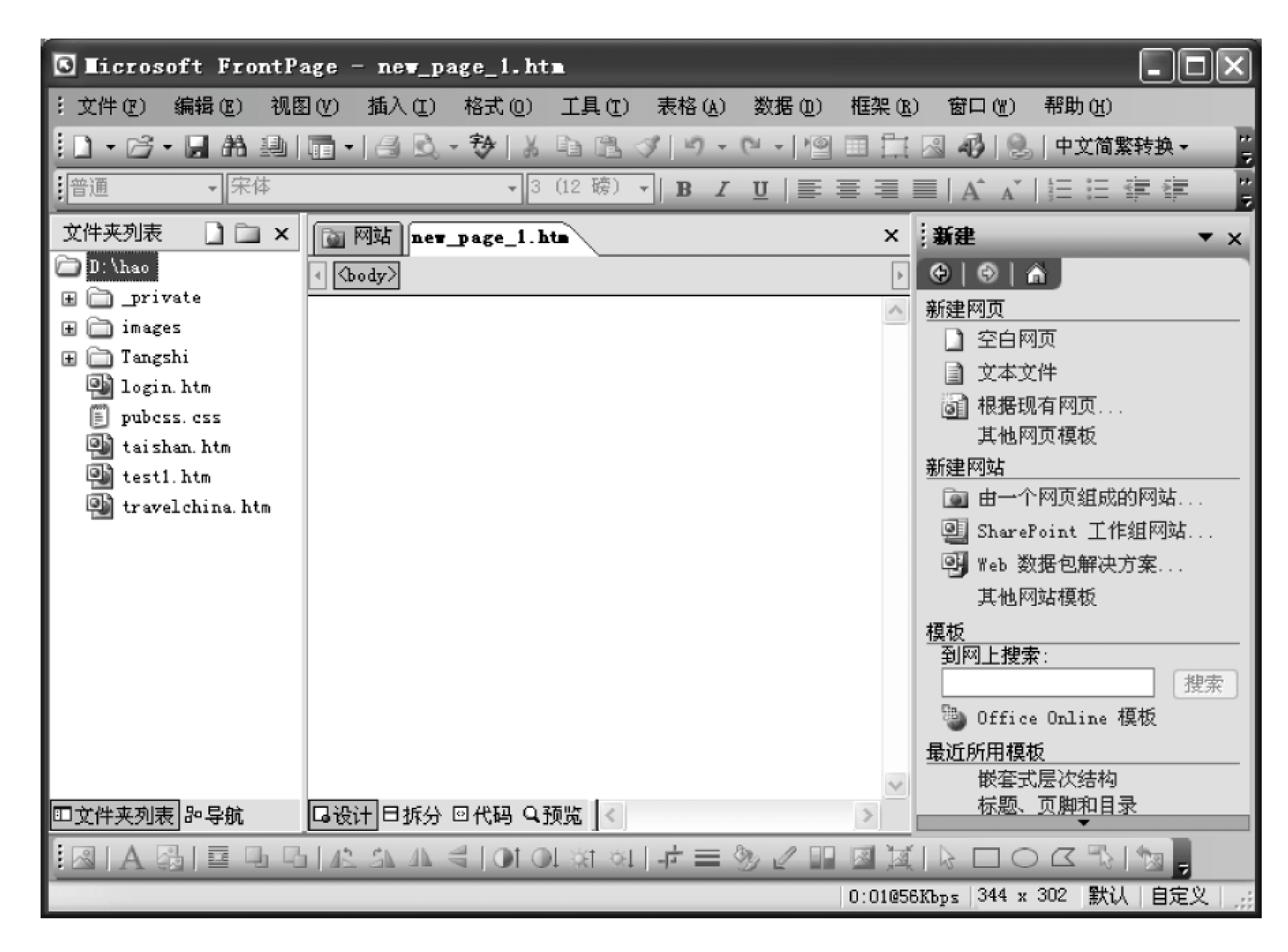


图 4-5　FrontPage 2003 主窗口

2. 拆分模式

选择拆分模式，则客户工作区分为上下两个区域，下面的区域显示设计的网页，上面部分显示对应的 HTML 代码。该模式的目的是将设计中的网页元素和对应代码准确定位，当单击设计视图的某个网页元素时，在上面的代码视图将自动定位到该元素对应的 HTML 代码，这对于一个较大的网页修改是非常方便的，便于手工调整。

3. 代码模式

在代码模式下，将显示设计中的网页对应的 HTML 代码。一般情况下，用户在设计模式下设计网页，然后在代码模式下查看网页对应的 HTML 代码。另外，用户也可以直接在该模式下编辑、修改其中的 HTML 代码，适用于那些对 HTML 比较熟悉的用户。如果网页中包含一些脚本程序，则这些脚本程序也需要通过代码模式进行编辑。

如果用户需要复制其他网页的 HTML 代码，应选择代码模式，将被复制的内容复制到当前网页的适当位置，而不应该在设计模式下进行复制。

4. 预览模式

选择预览模式，将对设计的网页进行预览，与通过浏览器所看到的网页一致。另外，如果网页中包含脚本程序，在程序的调试阶段通过 FrontPage 的预览模式可以很容易查出程序出错的位置和错误性质。

4.2.2 网站的新建与维护

FrontPage不仅是一个页面制作工具,同时,它还可以进行站点管理操作,包括创建、删除、打开、发布站点,以及进行站点的维护等工作。也可以对站点的文件和文件夹进行操作以及对站点进行安全性管理。

1. 新建网站

创建一个Web站点,本质上就是建立一个主目录,然后在主目录下创建子目录和网页文件。用户可以手工建立站点目录结构,通过资源管理器可以定位具体的文件夹和文件。为了管理方便,现在的网页制作工具都提供了新建站点和站点管理,以方便对站点的维护。

在FrontPage中,提供了创建站点的模板和向导,通过它们可以很容易地建立起一个新站点,即创建站点目录结构。操作方法是:选择"文件"→"新建"命令,显示"新建"任务窗格;在"新建网站"选项组中,单击其中的任何一个超链接,都打开"网站模板"对话框,如图4-6所示。



图4-6 "网站模板"对话框

单击"个人网站",在"指定新网站的位置"文本框中,输入一个文件夹名,该文件夹即为站点主目录,单击"确定"按钮。向导则自动创建一个站点主目录,并根据模板,创建相应的子文件夹和网页文件,如图4-7所示。

在屏幕的左边,显示网站的目录结构,这便于站点的管理和维护。如果不显示文件夹列表,选择"视图"→"文件夹列表"命令,即可打开站点文件夹列表,以便于对站点的操作。在网站窗口的下部,显示"文件夹"、"远程网站"、"报表"、"导航"、"超链接"和"任务"多个选项卡,选择"导航"选项卡,显示网站中各个网页之间的超链接。

2. 本地网站维护

一个站点,本质上是由一个文件夹,以及其中包含的所有网页文件构成的。如果在



图 4-7　新建个人站点

Web 服务器上，一个站点是真实运行的网站。如果是在一台桌面机上，例如 Windows XP 上，同样可以创建一个站点，如果仅仅是纯 HTML 文件，不包含服务器脚本，它同样可以运行。此外，对于任意一个文件夹，用户也可以作为一个站点来打开，在其中新建网页。

因此，无论是在 Web 服务器上，还是在桌面机上，都可以使用 FrontPage 对一个本地的文件夹(看做网站主目录)进行维护，具体步骤如下。

(1) 在 FrontPage 中，选择“文件”→“打开网站”命令，打开“打开网站”对话框，如图 4-8 所示。



图 4-8　“打开网站”对话框

在“查找范围”下拉列表框中,选择网站对应的主文件夹,“网站名称”下拉列表框中不需要输入内容,单击“打开”按钮,则文件夹名即为网站名,然后显示网站视图,显示网站目录结构。

如果打开的文件夹是一个普通的文件夹,FrontPage 将在该文件夹下添加两个子文件夹_private 和 images。

(2) 选择“视图”→“文件夹列表”命令,显示网站文件夹结构。接下来就可以对站点进行各种各样的维护和管理操作了。

站点的维护主要是指:在站点主目录下新建、删除、修改文件夹;在某个文件夹中,新建网页、修改网页或删除网页文件等。在网站“文件夹列表”中,在某个文件夹上右击,打开快捷菜单,可以选择“删除”、“重命名”、“新建(文件夹、空白网页)”等命令。

如果在某个网页文件上双击,则打开该网页文件,显示网页设计视图,可以进行网页的制作(编辑)和内容修改等操作,即网页制作。

4.2.3 新建网页

一个网站对应一个主目录,里面包含大量的网页文件,这些网页通常按照网页功能组织在不同的文件夹中。因此,一个网站和传统的软件开发中的一个项目(Project)类似,每一个网页文件都是网站的一部分。一种良好的网页制作思想是,在新建或编辑网页以前,应该先打开一个站点,因为,任何网页都是一个站点的一部分,而不是孤立存在的。

在 FrontPage 2003 中,要新建一个网页,可以使用“文件”菜单命令或主窗口右侧的任务窗格导航完成(如果没有开启任务窗格,选择“视图”→“任务窗格”命令),通常经过以下几个步骤。

(1) 选择“打开网站”命令,选择要打开的网站主目录。因为每一个网页都是网站的一部分,因此,新建网页前,首先应打开网页隶属的网站。

(2) 选择“文件”→“新建”命令,在客户区右侧显示“新建”任务窗格。

(3) 在网站文件夹列表中,单击要新建网页的文件夹。

(4) 在“新建”任务窗格中,单击“空白网页”超链接,将新建一个空白网页。单击“其他网页模板”,则打开“网页模板”对话框,显示 FrontPage 提供的默认模板,如图 4-9 所示。



图 4-9 “网页模板”对话框(1)

用户可以用这些模板来建立相应的网页，在右下角的预览区域可以看到选中模板的外观。

新建网页后，系统将生成一个 html 文档内容的基本框架，在“代码”模式或“拆分”模式下，可以看到相应的 html 代码内容，如图 4-10 所示。

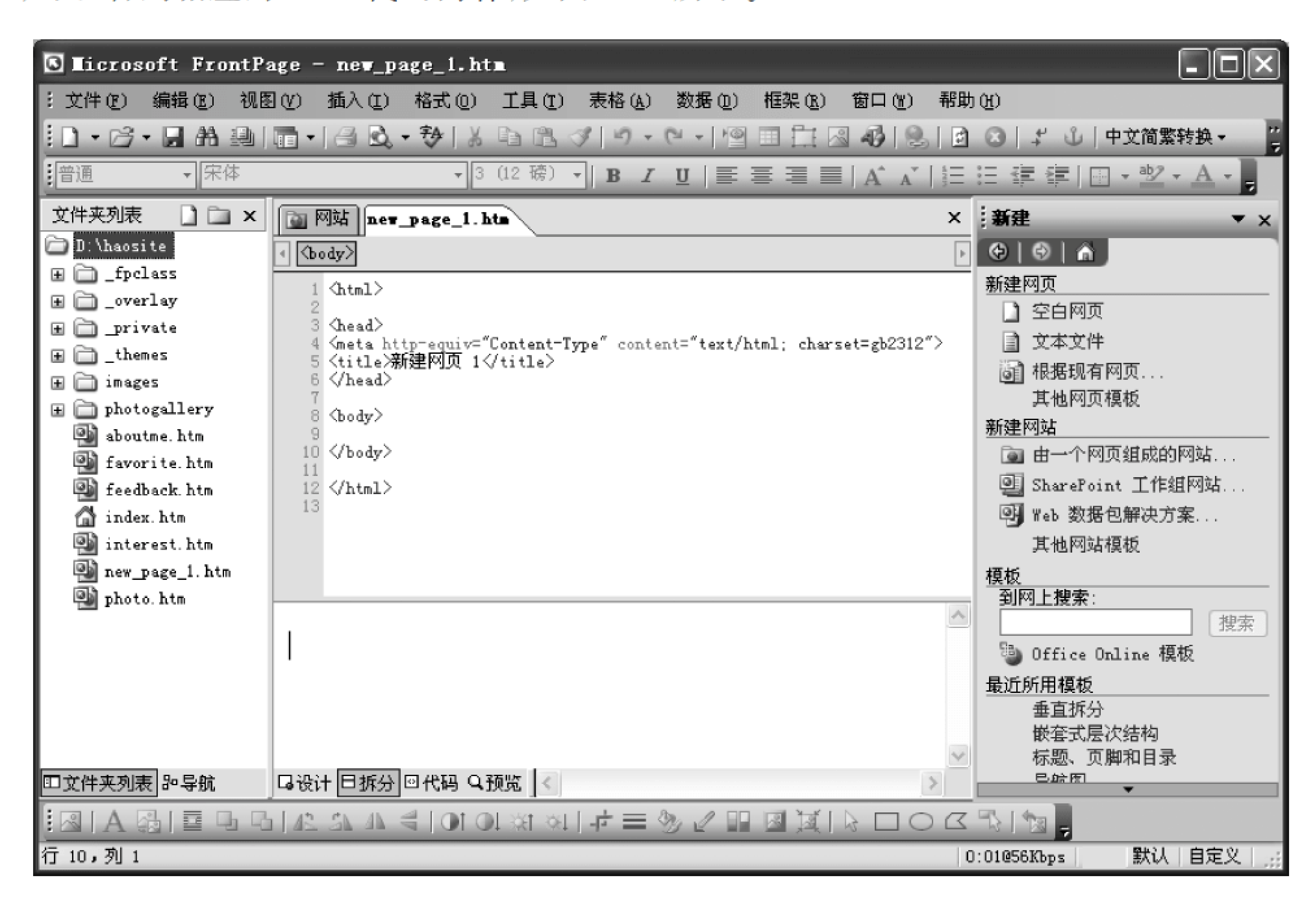


图 4-10　新建网页系统自动生成的 html 代码

可以看到，在文档的＜body＞…＜/body＞标记之间，还没有内容，具体内容根据用户的需要来设计。

使用 FrontPage 编辑网页，实际上就是利用 FrontPage 的菜单命令，在网页中输入文本，插入图片，建立超链接，插入表格、表单等操作，FrontPage 将自动生成对应的 html 代码，从而简化 html 文档的手工编写任务。

为了扩大网页编辑窗口的空间，可以将网站的“文件夹列表”视图关闭，如果需要打开，选择“视图”→“文件夹列表”命令（Alt＋F1）即可。

（5）网页编辑完成后，选择“文件”→“保存文件”命令，或者直接单击“常用”工具栏中的“保存文件”按钮，保存当前网页。

如果网页文件是第一次存盘，则打开“另存为”对话框，从中选择网页的存储位置、文件名和保存类型。

4.3　网页编辑

当网页文件创建后，接下来就是网页内容的编辑了。网页文件是一种符合 html 规范的纯文本文件，可以使用任意的文本编辑器进行编辑，但是这样的编辑需要非常熟悉 html 规

范，并且效率很低，容易出错。因此，对网页的编辑通常使用 FrontPage、Dreamweaver 等可视化的网页制作工具来完成。

使用 FrontPage 进行网页编辑，实际上就是利用 FrontPage 的菜单命令，在网页中输入文本，插入图片，建立超链接，插入表格、表单等内容，FrontPage 将自动生成对应的 html 代码，从而简化 html 文档的手工编写工作。

4.3.1 输入文本内容

文字是 Web 页的主要内容，在 html 规范中，有很多标记用于对文本进行标记，如<p>、<font>等，使用这些标记可以实现文本显示的格式化，即设定文本在浏览器中的显示效果。

在 FrontPage 中，文本的输入和格式化非常简单。在设计模式下，在编辑区域进行输入，再通过“格式”工具栏对文本进行格式化，或者通过“超链接”工具按钮将文本设置为超链接(<a>…标记)。

例如，在设计模式下输入一行文本“泰山”，并进行简单的格式化(居中，加粗，字号，颜色)，如图 4-11 所示。

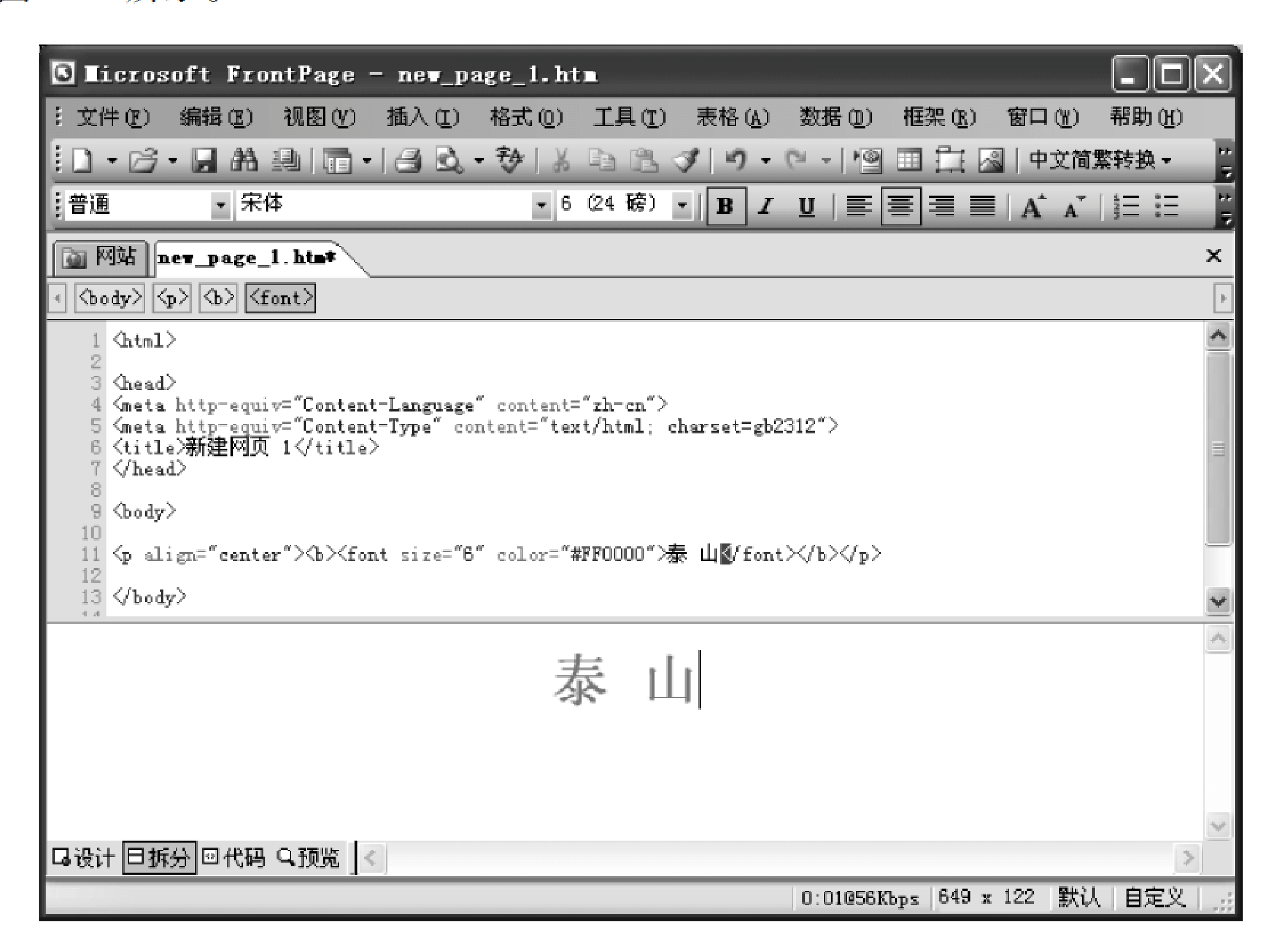


图 4-11 文本内容的输入与格式化

在拆分模式下，或代码模式下，可以看到在<body>…</body>标记内增加了如下代码：

```
<p align = "center"><b><font size = "6" color = "#FF0000">泰 山</font></b></p>
```

这段代码是 FrontPage 根据我们的输入和格式化自动生成的，可见通过 FrontPage 进行网页设计要比用“记事本”等程序手工书写 html 代码方便和快捷得多，并且也不容易出

错，不需要记忆各种标记属性，这正是 FrontPage、Dreamweaver 等工具的优势所在。

使用 FrontPage、Dreamweaver 等工具，用户也可以从记忆大量的 html 标记的任务中解脱出来，从而把重点放在网页布局的设计和业务逻辑的设计上。

4.3.2　插入图片

在网页制作中经常需要插入一些图片，以达到丰富信息内容和美化页面的双重功能。在网页中插入图片，即通过<img>标记来指向一个图片文件，为了页面布局的需要，通常还需要设置<img>标记的属性。

使用 FrontPage 在网页中插入一幅图片，具体操作步骤如下。

(1) 将插入点定位到要插入图片的地方(本例定位到文字的开始)。

(2) 选择“插入”→“图片”→“来自文件”命令，打开“图片”对话框；选择要插入的图片文件，单击“插入”按钮，则图片自动以无环绕样式的方式插入到网页中。

(3) 设置图片属性。由于图片的高度和文字不一致，接下来应该设置图片的环绕方式。在 FrontPage 2003 中，右击图片，在快捷菜单中选择“图片属性”命令，打开图片属性对话框，从中可设置图片环绕样式。

设置完成后的页面如图 4-12 所示。

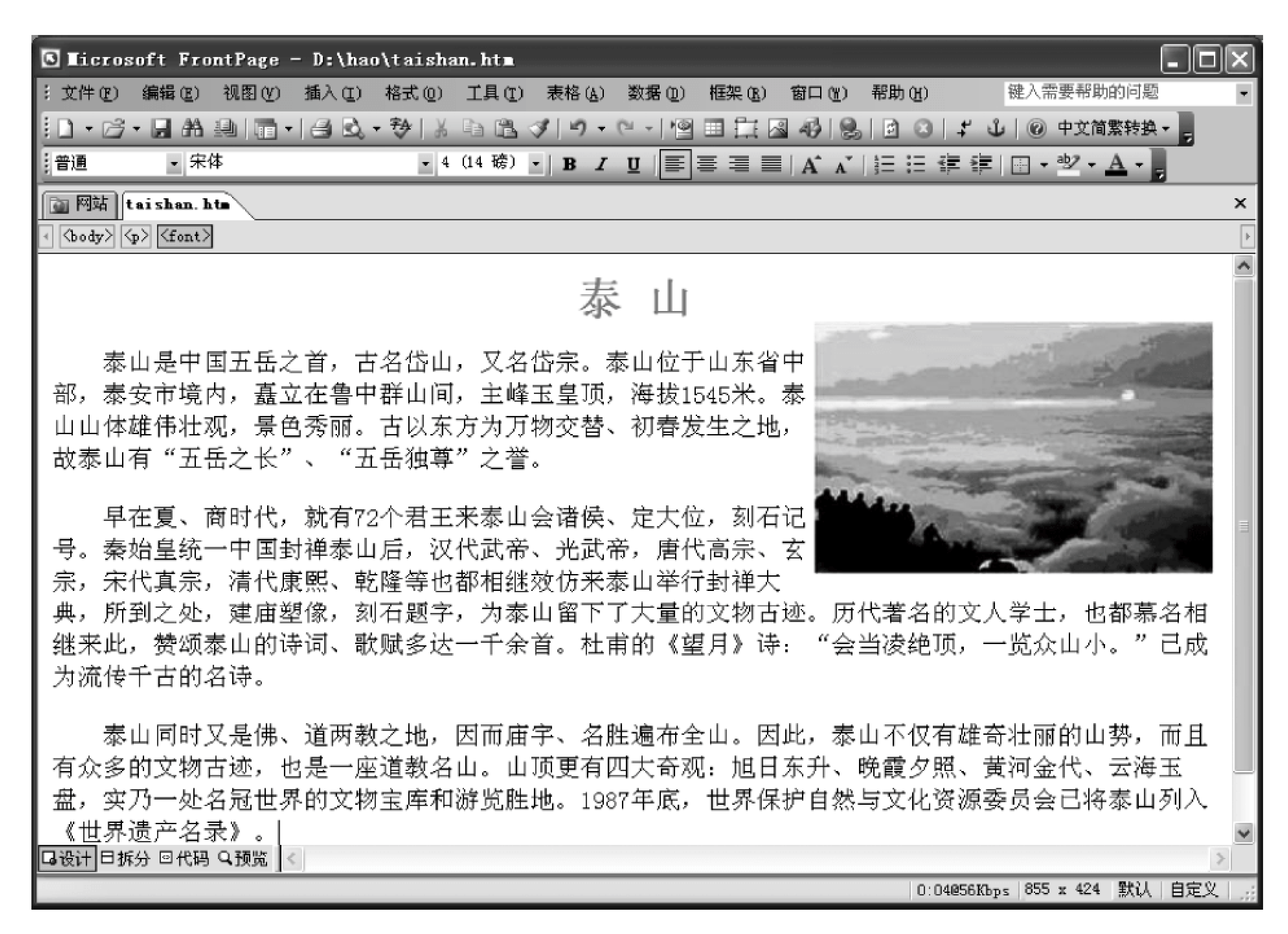


图 4-12　在网页中插入图片

选择“拆分”或“代码”显示模式，可以看到在<body>…</body>标记内增加了一行类似下面的代码：

```
< img src = "file:///c:/xxx/tai.jpg" width = "281" height = "175" align = "right">
```

这是图片的绝对路径,在Web应用开发中,要避免使用绝对路径。因为,当网页文件或者存储图片的目录发生改变,在打开网页时就会出现找不到图片文件的错误,图片在浏览器中不能被显示。

为什么会产生上述问题呢?这是因为:如果是在一个新建的网页中插入图片,因为当前网页尚未保存,因此无法判断网页文件和图片文件之间的相对路径,只能采用绝对路径的方式插入。此时,如果将新建的网页先保存一次,再插入图片时,在html代码中就会使用相对路径。或者,即使不是先插入了图片,只要保存一次html文件,原先的绝对路径也会变成相对路径。但是,如果html文件和图片不在一个驱动器上,将不能转换成相对路径。

最后要记住,既然是一个Web应用,所有的网页和用到的图片以及其他各种文件都应该保存在Web站点的主目录下,或者主目录下面的子目录下。所有这些文件共同构成了一个Web应用,所有的引用都应该是相对路径,这样当Web应用,即该站点被复制到其他的目标位置时,才不会出现找不到网页或图片的错误。因此,如果页面编辑前执行了"打开网站"命令,当插入的图片不在主目录下,保存网页时会出现"保存嵌入式文件"提示对话框。

图片插入以后它在页面上的大小、位置等可能并不理想。要使得图片符合用户的要求,需通过设置图片属性来完成。图片属性的设置分别对应了<img>标记的不同属性,用户可以通过"拆分"或"代码"视图查看生成的HTML代码。此外,将图片直接插入网页中,要进行精细的布局是很困难的,要对网页进行布局,通常需要使用表格来完成。

4.3.3 建立超链接或书签

使用FrontPage在网页中建立超链接非常简单,只需要选定超链接的文本或图片,然后,单击工具栏中的"超链接"按钮,即可打开"插入超链接"对话框,从中输入被链接的URL即可。

1. 定义书签

书签是文档中的一个特定标记,便于打开网页定位到特定的位置。使用FrontPage在网页中定义书签,可按照下面步骤操作:

在网页中选定文字,选择"插入"→"书签"命令,打开"书签"对话框;在"书签"对话框中,显示书签名称和网页中已经定义的书签列表,输入书签名,单击"确定"按钮,即可定义一个书签。生成的html代码形式如下:

```
<a name = "书签名">书签文本</a>
```

在定义书签时,可以不选中任何文本,这样只是在插入点处定义一个书签。

2. 文本超链接

使用FrontPage在网页中建立文本超链接的操作步骤如下。

(1) 选择要建立超链接的文本。

(2) 选择"插入"→"超链接"命令,或直接单击常用工具栏中的"超链接"按钮,打开"插入超链接"对话框,如图4-13所示。

(3) 在"链接到"列表框中,有四种主要的链接到目标类型。

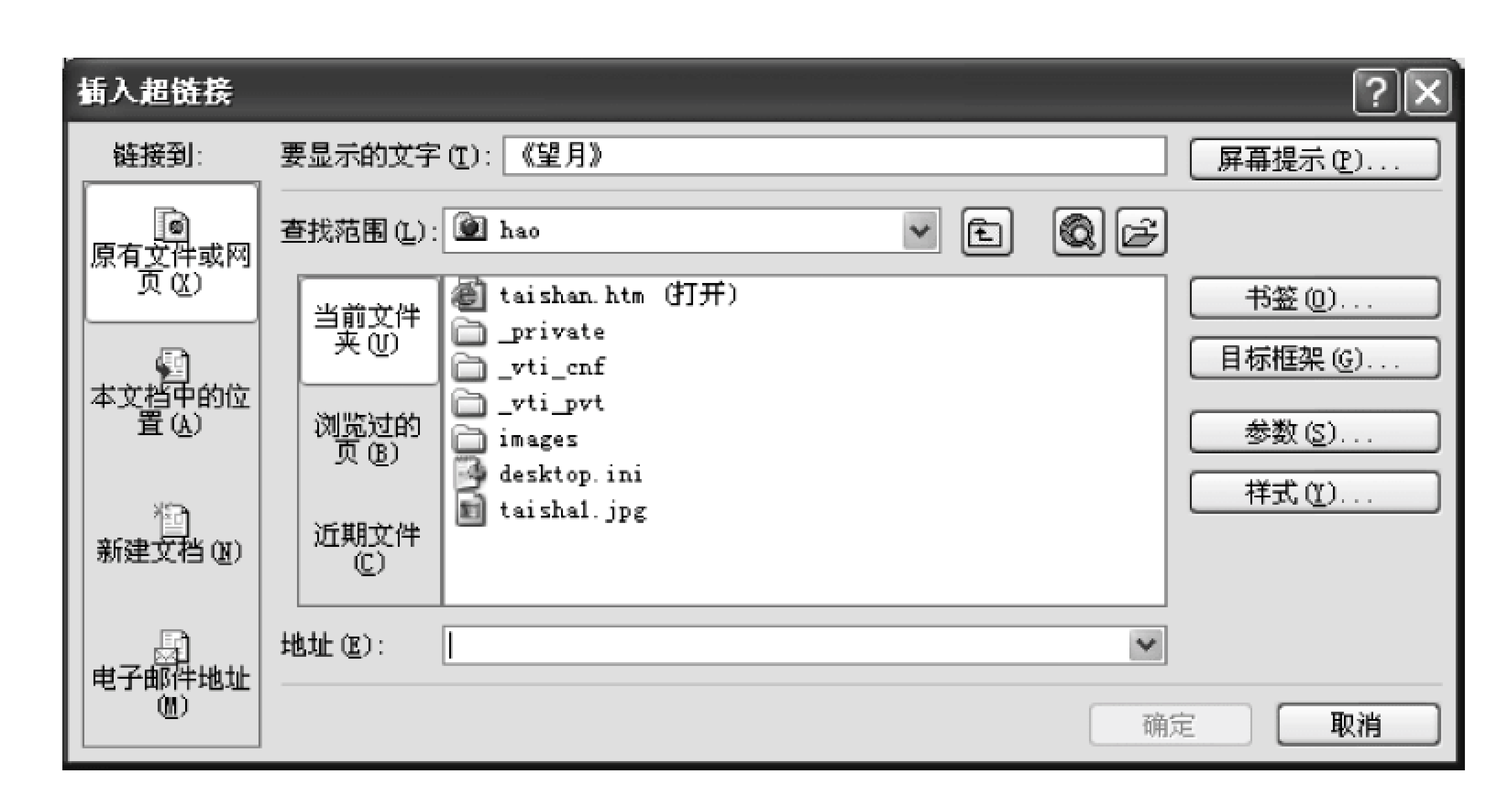


图 4-13 “插入超链接”对话框

① 原有文件或网页：通过 Web 浏览器查找和选择被链接的网页。

② 本文档中的位置：链接到网页中的一个书签，通过“选择文件”对话框在本地机上查找和选择被链接的文档。需注意文件和被链接的 html 文档之间的路径问题。

如果是同一个 Web 应用中的 Web 页面，必须使用相对路径，方法类似于插入图片。如果是当前 Web 应用以外的 Web 页面，需要给出完整的 URL 地址。

③ 新建文档：新建一个空白文档并且与之建立超链接。

④ 电子邮件地址：与一个 E-mail 地址建立超链接。

(4) 设置目标框架和书签。单击右侧的“书签”按钮，可以显示选择的网页中定义的书签。单击右侧的“目标框架”按钮，可以指定单击超链接时打开网页的窗口，分别对应<a>标记中的 target 属性和 name 属性。

(5) 通过“样式”按钮，可以设置<a>标记的用户自定义样式类 class 属性和 ID 属性。

当上述设置完成后，单击“确定”按钮，完成超链接的建立，即建立一个文本超链接。自动生成对应的 html 代码，形式如下：

```
<a href = "poems/wangyue.htm" target = "_blank" >«望月»</a>
```

当文本对应一个超链接时，文本将以特殊的颜色显示，超链接文本默认显示为蓝色带下划线的文字。显示网页时，当鼠标指针指向该文本时，指针将变为手形。

3. 图片超链接

使用 FrontPage 也可以为图片建立超链接，具体操作步骤如下：

选中将要建立超链接的图像；选择“插入”→“超链接”命令，打开“超链接”对话框，接下来的设置与文本超链接的操作相同。

4.3.4 图像地图

在 HTML 规范中，图像地图是一种常用的页面导航界面，对于热点的边界点(x,y)坐标，手工定义几乎不可能，因此，在页面中使用图像地图都是通过 FrontPage 等页面制作工具完成的。使用 FrontPage 建立图像地图的具体步骤如下。

(1) 选择“插入”→“图片”→“来自文件”命令，插入一幅图片。

(2) 单击图片，则在FrontPage窗口的底部显示图片工具栏。如果不显示图片工具栏，可在图片上右击，在快捷菜单中选择“显示图片工具栏”命令，即可显示“图片”工具栏，如图4-14所示。

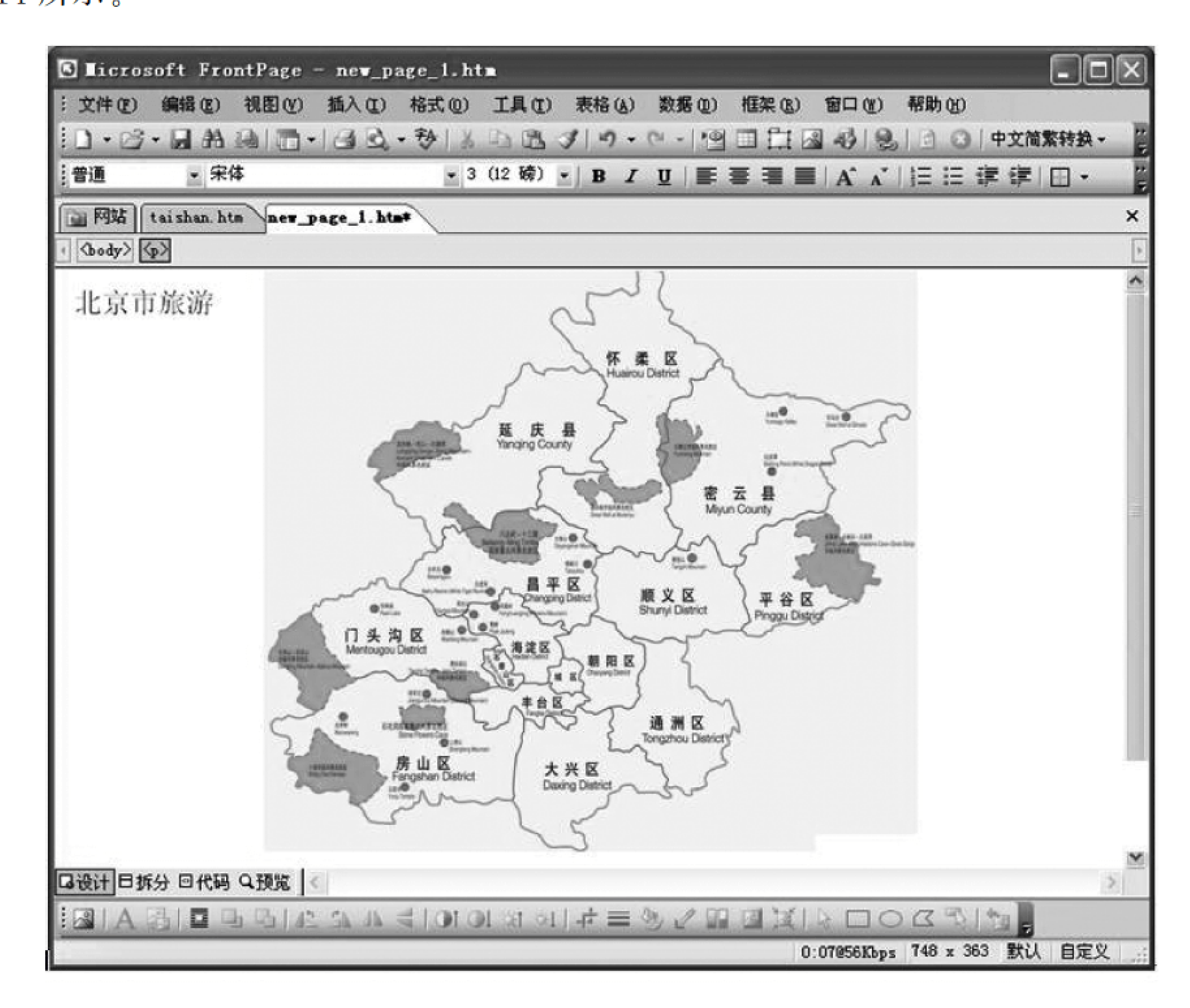


图4-14 定义图像地图

定义图像的热点可通过“图片”工具栏完成，其中有4个定义热点的工具：长方形热点、圆形热点、多边形热点和突出显示热点。

定义热点的方法很简单：选中图像，这时上述的4个工具变为可用；然后按住鼠标左键在图像上的目标区域拖动即可。对于多边形热点，则可以在逐个顶点位置单击，直到形成一个封闭的区域为止。热点区域建立后，将自动打开“超链接”对话框，从中输入要链接的URL即可。

网页显示时，图像上的热点区域是不可见的，但是在热点设置时按下“图片”工具栏中的“突出显示热点”按钮，这样在显示网页时，如果单击热点区域，可以看到区域的外框。

自动生成如下所示的html代码：

```
<map name = "FPMap0">
<area href = "city/beijing.htm" shape = "circle" coords = "286, 100, 17">
<area href = "city/jinan.htm" shape = "polygon" coords = "279, 129, 295, 111, 323, 122, 305,
136, 287, 145">
</map>
<img border = "0" src = "chinamap.gif" width = "433" height = "270" usemap = "#FPMap0">
```

4.3.5 插入表格

在网页设计中，表格不仅用来显示数据，而且还是进行网页布局的重要手段。使用表格，可对各种网页元素的位置进行精确控制，例如文字和图片混排，从而使设计的网页布局整齐、美观。

使用 FrontPage，在网页中插入表格，可以按照下面的步骤操作。

(1) 在网页中插入表格。选择菜单“表格”→“插入”命令，打开“插入表格”对话框，从中可输入要插入表格的行数和列数，设置布局、边框、背景等属性，即可插入一个表格。

(2) 表格编辑。插入到网页上的表格，可以进行一系列的编辑操作，如：调整行高和列宽，单元格的合并与拆分，删除行、列或单元格，插入行、列或单元格等。

(3) 设置表格属性。将插入点放在表格当中，右击，从快捷菜单中选择“表格属性”命令，打开“表格属性”对话框，如图 4-15 所示。

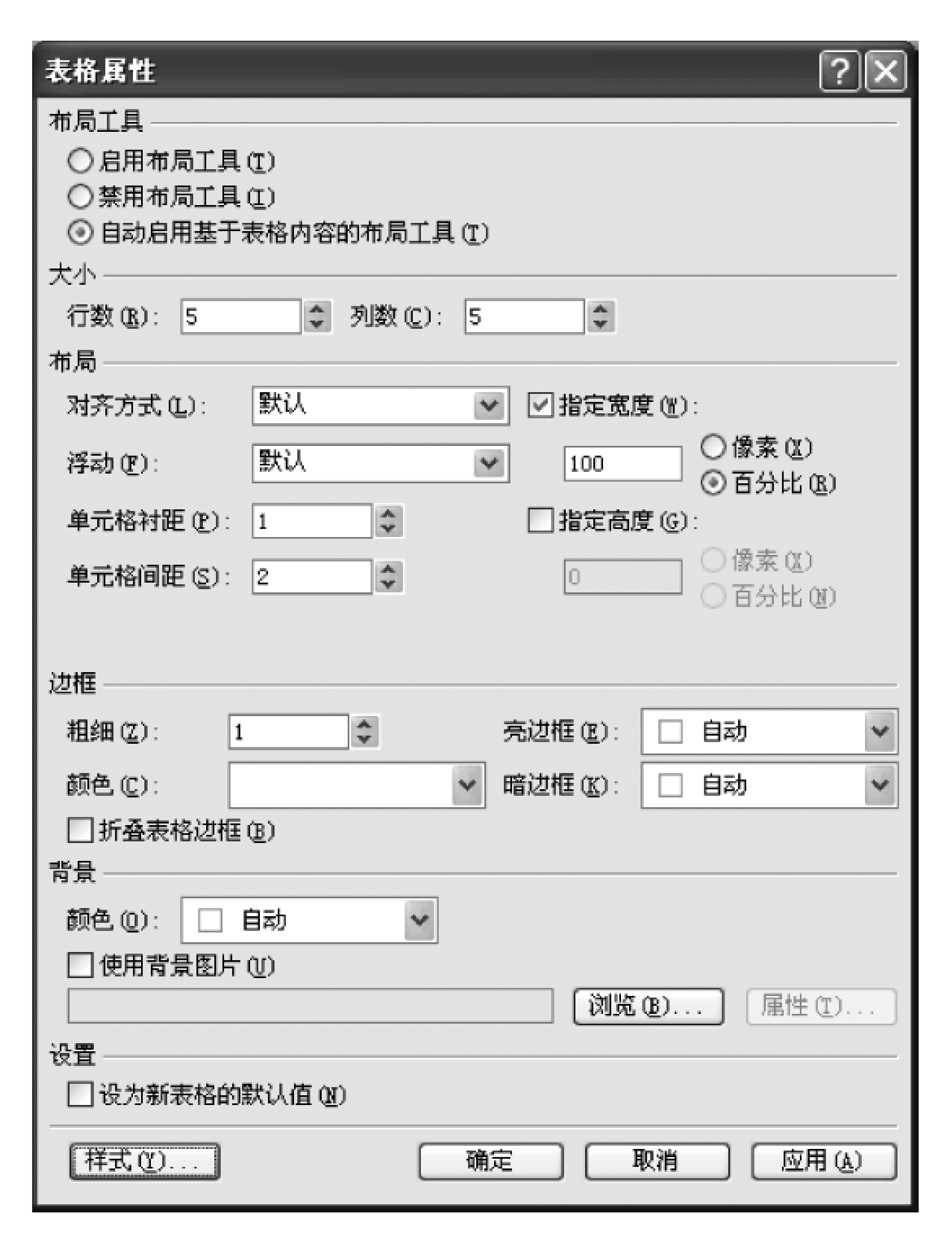


图 4-15 “表格属性”对话框

在“表格属性”对话框可进行以下几个方面的设置。

① 布局属性：包括表格的对齐方式、表格宽度、单元格间距(cellspacing)、单元格衬距(cellpadding)。其中，单元格间距指单元格之间的距离，默认值为 2，可设值为 0；单元格衬距指单元格中文字到表格线的距离，默认值为 1。

② 表格边框设置，设置表格边框的粗细和颜色。

③ 表格背景，可以设置表格所用的背景色或背景图片。

在网页中,如果要使表格随着浏览器的窗口变化而变化,在“表格属性”对话框中,选中“指定宽度”复选框,然后选择“百分比”单选按钮。此时,如果表格的宽度大于浏览器窗口的宽度时,表格内容将换行,可能影响表格的外观。如果在“表格属性”对话框中,选中“指定宽度”复选框,并且选择“像素”单选按钮,则表格的显示将不随浏览器窗口的变化而变化。

(4) 设置单元格属性。当需要对单元格进行特殊设置时,将插入点放到单元格中,或者选定要设置的单元格,右击,从快捷菜单中选择“单元格属性”命令,打开“单元格属性”对话框,从中可以对一个单元格进行设置,这些设置将影响<td>标记的属性。

通过 FrontPage 的“拆分”视图或“代码”视图,可以看到生成的 html 代码,表格属性和单元格属性分别对应<table>标记、<tr>标记以及<td>标记的有关属性。

需要特别说明的是,在图 4-15 所示的“表格属性”对话框中,单击左下角的“样式”按钮,可以设置<table>标记的样式类 class 属性和 id 属性。如果网页中包含 css 文件,用户可以选择其中的用户自定义样式类。使用样式类(class 属性)来定制 table 的外观是一种良好的页面制作习惯,它便于对网页的修改,但是这却是许多开发人员忽视的一点。

4.3.6 插入表单

对于一个负责用户交互的网页,其主要内容就是表单。使用 FrontPage 在网页中插入表单,具体操作步骤如下。

(1) 选择“插入”→“表单”→“表单”命令,插入一个表单,然后依次执行要插入的控件(表单元素)命令,以便在表单中插入需要的表单元素。

(2) 调整表单布局。表单一般和表格联合使用,通过表格设置表单布局。FrontPage 有时候使的这些标记交叉,为此,可以在“拆分”或“代码”视图中,手工调整代码。例如,将<form></form>标记对调整到<table></table>标记对的外面。

在使用 FrontPage 插入表单或控件时,如果插入了多个<form>标记,也可以手工删除,只保留一个<form></form>标记对,手工把所有的控件标记都移动到<form></form>内部。

(3) 设置表单属性。在表单上右击,在快捷菜单中选择“表单属性”命令,打开“表单属性”对话框,如图 4-16 所示。



图 4-16 设置表单属性

在“表单属性”对话框中，可以进行各种设置，即对应<form>标记的属性设置。

（4）设置表单域属性。在表单域上右击，在快捷菜单中选择“表单域属性”命令，打开“表单域属性”对话框。不同类型的表单域，其属性对话框也不相同，文本框控件对应的表单域属性对话框如图 4-17 所示。

文本框属性
名称(N)：myaccount
初始值(A)：guest
宽度(W)：20　Tab 键次序(T)：1
密码域：○是(Y)　◉否(O)
样式(S)...　验证有效性(V)...　确定　取消

图 4-17　设置表单域属性

在“表单域属性”对话框中，输入相应的属性值，为脚本编程需要，一般要设置表单域名称等属性，最后单击“确定”按钮。

当表单和表单域属性设置结束后，切换到“代码”模式或“拆分”模式，可以看到生成的 html 代码，如图 4-18 所示。

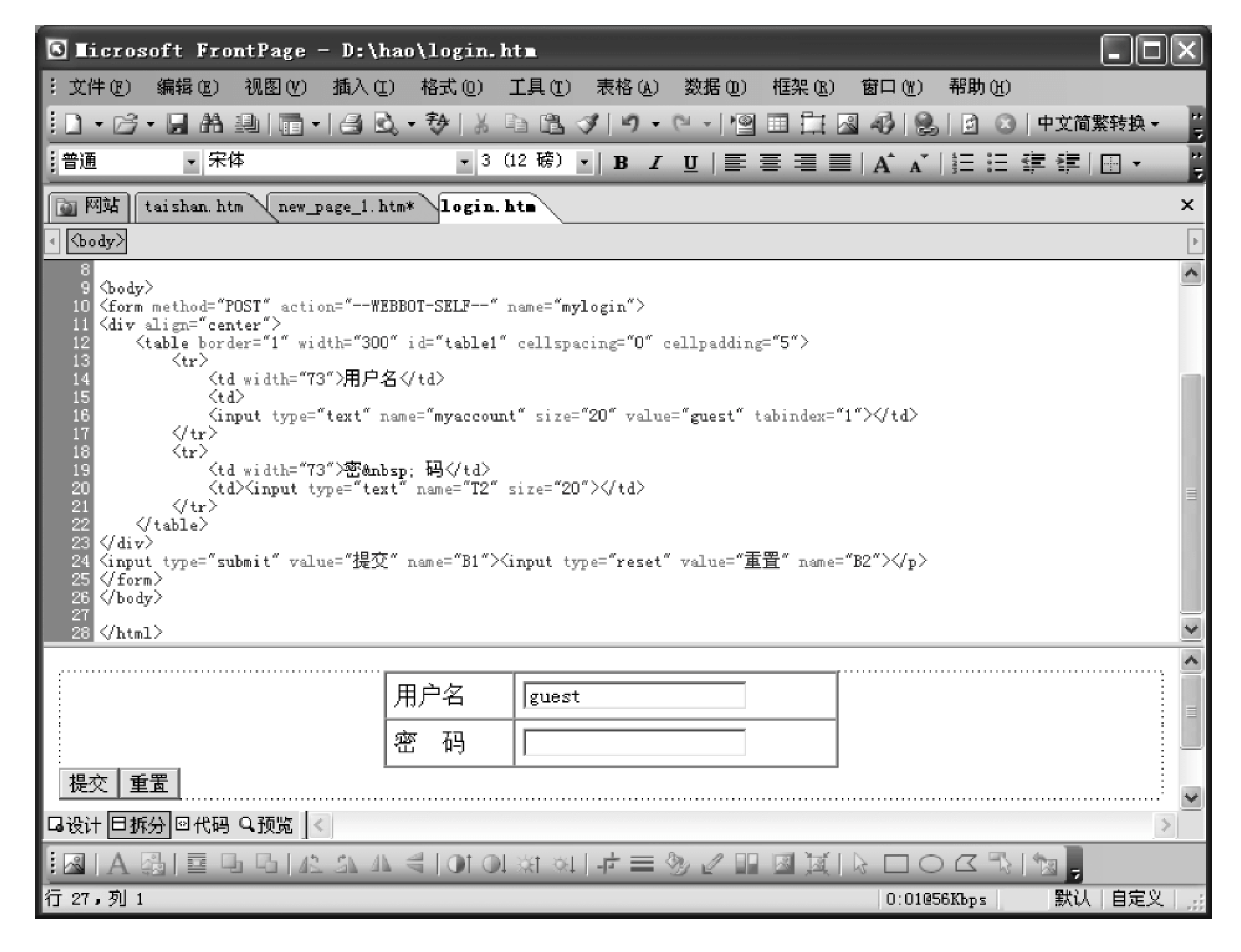


图 4-18　插入表单及其属性设置

4.3.7 插入图层

图层是 HTML 中重要的标记,在 FrontPage 中,选择设计视图,在工具栏中单击“插入层”按钮,即可通过鼠标在页面上创建一个图层。在图层上右击,打开快捷菜单,选择“层属性”命令,在右侧显示“层”面板,从中可以进行图层的边框和底纹、定位、行为等设计,如图 4-19 所示。

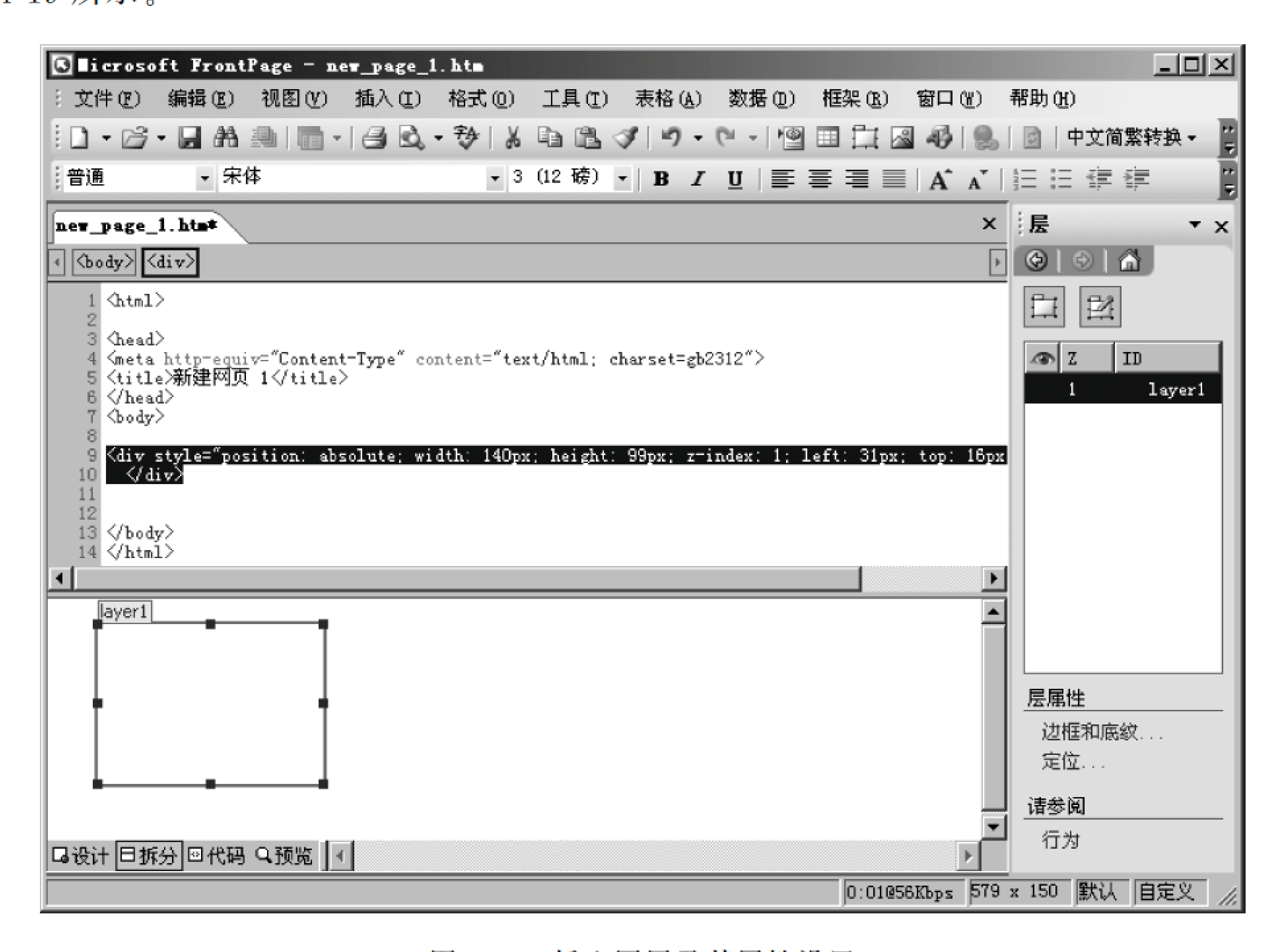


图 4-19 插入图层及其属性设置

4.4 设置标记属性

在 HTML 规范中,每个标记都含有不同的属性,通过设置标记属性可以修改标记的默认显示样式。标记的属性众多,很难记忆。利用 FrontPage,在“设计”视图下,通过一些标记的属性对话框可以进行标记的属性设置。除此之外,还可以在“代码”视图下,通过 FrontPage 的智能感知技术设置标记属性。

4.4.1 使用属性对话框

在现代软件中,对于每一个对象,无论是操作系统级的对象(如文件夹、文件等),还是软件系统中的对象,例如 FrontPage 设计视图下的各个标记(例如表格、单元格、表单、输入域),还是图层,右击一个对象,都会弹出一个快捷菜单,里面包含了当前对象的常用操作命令,这就大大地方便了对象的操作。

在 FrontPage 中，设置标记属性，可以在设计视图中，右击一个设计对象，在快捷菜单中选择“属性”命令，在弹出的属性对话框中进行设置，这种设置将修改代码视图中生成的代码，即修改标记的属性。

4.4.2　IntelliSense 技术

现在大多数的开发环境都使用智能感知(IntelliSense)技术为用户提供帮助。所谓 IntelliSense，是指当用户编辑到一个对象时，系统能动态地显示当前对象的方法、属性名列表，从而保证用户输入的正确性，或从中选择输入。在 Boland 的开发环境中，类似的技术称为 CodeInsight。下面举例说明。

在一个网页中，假定要设置一个超链接的属性。选择“代码”视图，将插入点定位到<a>标记，按空格键，则自动打开一个窗口，显示<a>标记的所有属性，包括一般属性和事件属性，如图 4-20 所示。

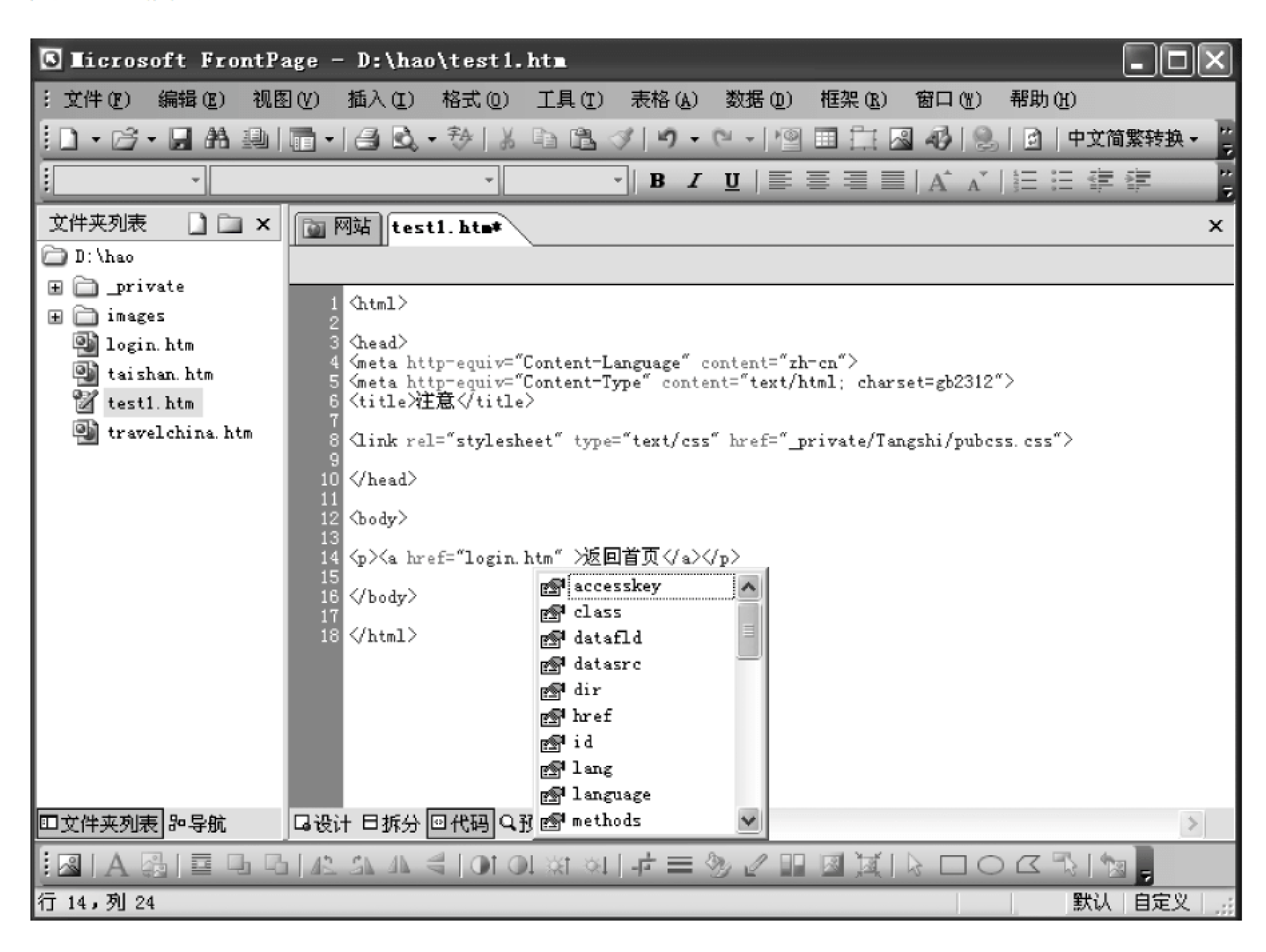


图 4-20　使用 IntelliSense 技术

现在大多数软件都使用了 IntelliSense 技术，有的软件默认情况下不打开智能感知，需要对软件进行设置，开启 IntelliSense 功能。

4.4.3　使用行为面板

标记中的事件属性对应的事件，又称为行为，即用户对所标记对象的操作行为。每个行为事件往往都对应一个函数，即标记的事件属性取值通常是一个函数。在 FrontPage 中，有许多内置的行为函数，使用这些函数可以简化用户的编码。具体操作如下。

(1) 要设置标记的行为,首先选择“设计”视图,选择“格式”→“行为”命令,打开“行为”任务面板。

(2) 单击“插入”按钮,在下拉列表中,可以选择一个行为。则在下面的事件列表中,显示行为对应的默认事件。

(3) 如果需要修改行为对应的事件,可以单击事件,则显示当前对象的可选事件列表,选择一个事件作为当前行为的激活事件即可。

执行以上操作步骤,结果如图 4-21 所示。

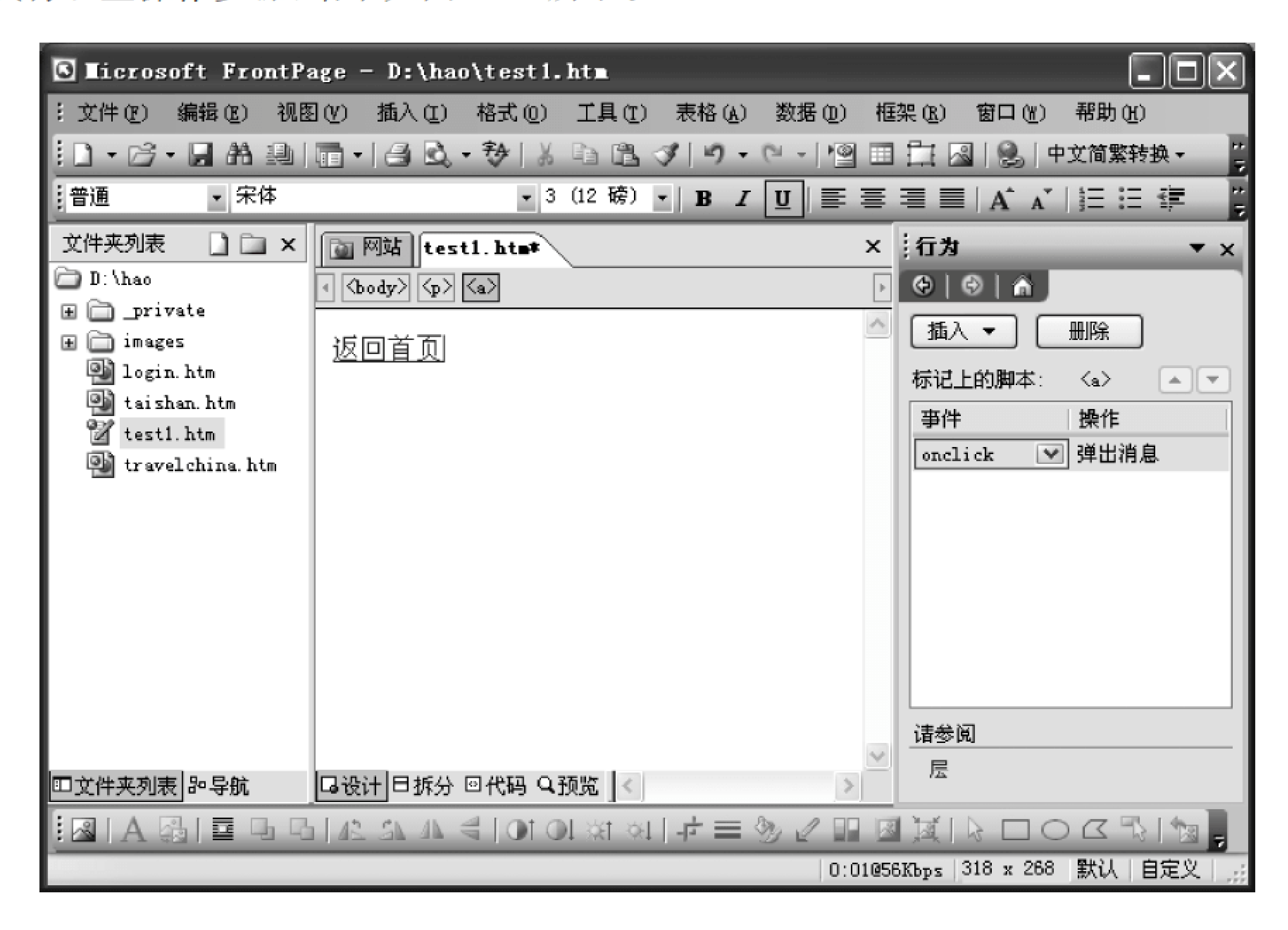


图 4-21 设置标记的事件属性

转到“代码”视图,可以看到标记中增加了相应的事件属性,并可以看到自动生成的脚本程序代码。

4.5 定义和使用样式

HTML 允许用户在标记中使用 class 属性修改标记的默认显示样式,手工定义样式需要记住很多属性及取值,非常麻烦。在 FrontPage 中,提供了修改样式和新建样式类的功能。为了在多个页面中实现样式的共享,可以将用户自定义样式存储为样式表文件。

4.5.1 定义样式

如果要在一个网页中使用样式,在 FrontPage 中,可以通过选择“格式”→“样式”命令来实现,具体操作步骤如下。

(1) 选择“格式”→“样式”命令,打开“样式”对话框,如图 4-22 所示。

(2) 在“列表”下拉列表框中,可以选择“用户自定义样式”或者“HTML 标记”。



图 4-22　“样式”对话框

① 修改 html 标记默认样式。选择“HTML 标记”，则意味着修改默认样式，列表中将列出 html 规范的所有标记，然后选择一个样式，单击“修改”按钮，修改所选的样式。

② 新建用户样式类。选择“用户自定义样式”，单击“新建”按钮，可以创建一个新的样式类，并对样式进行字体等属性的设置。

此时在 HTML 视图中，将看到文档的<head>…</head>内部生成类似如下的代码：

```
<style>
a:active     { font - size:10pt; color: #0000FF }
a:hover      { font - size: 10px; color: #FF0000 }
.warning     {font - family: 宋体; font - size: 8pt; color: #FF0000; font - weight: bold }
</style>
```

用户自定义样式后，可以在 FrontPage“格式”工具栏左边的样式列表中显示用户自定义的样式，用户可以用自定义样式格式化网页中的文档内容。此时，在生成的 HTML 代码中看到相应标记内增加了 class 属性，例如“<p class="warning">注意事项：</p>”。

4.5.2　使用样式表文件

在网页内定义的样式，只是用于网页内部。为了实现多个网页之间一致的外观和自定义样式的共享，通常需要将样式定义保存为.css 文件，这样多个网页就可以共用样式，保证网站中网页风格的一致性。

建立样式表文件(.css 文件)有两种方法。

(1) 选择“文件”→“新建”命令，打开“新建”任务窗格；单击“其他网页模板”超链接，打开“网页模板”对话框(图 4-9)；选择“样式表”选项卡，然后选择一个样式表模板，从而建立一个样式表文件。接下来，通过选择“格式”→“样式”命令来编辑样式即可。

(2) 新建一个普通的页面文件，自定义样式完成后，删除<style>…</style>标记对本身及以外的所有 html 代码，只保留<style>…</style>标记对内部的样式定义部分，

然后选择“另存为”命令，选择保存类型为“css 文件(.css)”。

当样式表文件定义完成后，在其他网页中就可以使用其中的样式了。选择“格式”→“样式表链接”命令，打开“链接样式表”对话框；选择一个.css 文件，例如 mystyles1.css，则在该网页的<head>…</head>内增加下述语句：

```
<link type = "text/css" rel = "stylesheet" href = "mystyles1.css">
```

注意，如果新建一个网页文件，在没有保存前，“格式”→“样式表链接”命令是灰化的，无法使用。这是因为，网页在存盘前，无法确定网页文件和要引用的 css 文件的相对路径，因此不能使用 css 文件。此时，只要保存一下网页文件，然后“样式表链接”命令就可以使用了。

4.6　Frame

在网页布局设计中，框架网页是经常使用的一种网页类型，它可以把浏览窗口分成几个区域，每个区域可以显示一个不同的页面，即对应一个单独的网页文件。

4.6.1　新建框架网页

在 FrontPage 2003 中创建框架网页可以通过专门的框架网页模板完成，方法如下。

(1) 选择“文件”→“新建”命令，打开“新建”任务窗格；在新建网页区域，单击“其他网页模板”超链接，打开“网页模板”对话框，如图 4-23 所示。



图 4-23　“网页模板”对话框(2)

(2) 选择“框架网页”选项卡，则在左侧列表框中显示多个框架网页模板的图标，单击其中之一，可以在右侧的预览区显示该模板的布局样式。

(3) 双击选中的模板，或者选中模板后单击“确定”按钮，即可建立相应的框架网页，

图 4-24 就是选中“嵌套式层次结构”后，所建框架网页的初始状态。



图 4-24 新建框架网页示例

在这个框架网页中，浏览窗口被分成了 3 个框架，各个窗格显示不同的页面。可以看到，初始状态下每个框架中有两个按钮。

① 设置初始网页。所谓初始网页，是指浏览器第一次载入框架网页时该框架所显示的网页。该按钮就是设置一个已经存在的网页作为该框架的初始网页。单击时可以打开“创建超链接”对话框，选择一个网页文件或输入一个网页的 URL，然后单击“确定”按钮，即可以在当前框架中打开所超链接的网页文档。

② 新建网页。单击该按钮时，可以新建一个普通的网页(空白页)作为初始网页。

当新建或打开一个框架网页后，FrontPage 窗口下方的显示模式选项标签将发生以下变化：

(1) 增加了“无框架”标签。当浏览器不支持框架网页时所要显示的内容。默认情况下将显示文本“此网页使用了框架，但您的浏览器不支持框架”。

(2) 当前网页为框架网页时，“框架”菜单中的命令将变为可用，主要包括“拆分框架”、“删除框架”、“框架属性”等命令。

4.6.2 对框架的常用操作

当框架建立后，可以对框架进行以下操作。

- 拆分与删除框架。在设计视图，单击框架，在“框架”菜单中选择相应的命令。
- 改变框架窗格的大小。将鼠标指针指向框架，当指针变成 ↔ 形或者 ↕ 形时，按住左键拖动，达到满意的位置后松开。
- 设置框架属性。鼠标指针指向框架，右击，从快捷菜单中选择“框架属性”命令，打开“框架属性”对话框，从中进行相应的设置。

本章小结

本章首先讨论了网页设计的概念，将网页设计分为面向业务逻辑的功能性设计(交互设计)和面向用户体验的视觉设计两个不同的层面。然后重点讨论了面向用户体验的页面布局设计、页面视觉设计和页面效果设计的有关问题。然后对 FrontPage 工具的功能和使用进行了较详细的讲解，通过 FrontPage 的各菜单命令的使用，来展示了网页制作工具在网页制作上的作用，即自动生成 html 代码，从而提高了手工编写代码的效率。同时还介绍了强大的 IntelliSense 技术的使用。最后介绍了在 FrontPage 中创建和维护 css 样式表的方法，以及框架和浮动框架的使用。

习题 4

一、简答题

1. 在进行页面布局设计时，需要考虑哪些因素？请上网进行搜索，总结一下有哪些常用的网页布局。

2. 什么是页面视觉设计？有哪些主要的内容？

3. 简要说明页面设计的主要步骤。在页面效果设计中，为什么要进行切图？

4. 要对 Windows 平台中的网站进行远程管理，应如何设置？

5. 对于 FrontPage 工具，回答下列问题：

(1) FrontPage 的核心功能是什么？

(2) 在 FrontPage 中，设计者设计“任务窗格”的初衷是什么？如何打开或关闭？

(3) 在 FrontPage 的“文件”菜单中的“打开网站”命令的功能是什么？

(4) 在 Web 开发中，网页制作的流程应该是：先打开网站，然后新建网页或对已有网页进行编辑。这样的一个流程反映了什么思想？

(5) 在 FrontPage 中，可以在不打开网站的情况下，直接新建一个网页吗？

(6) FrontPage 设计“拆分”视图的目的是什么？

6. 怎样在 HTML 文档中插入图片？<img>标记的常用属性有哪些？怎样将这幅图片放在居中位置？

7. FrontPage 中不同的显示模式各有什么用途？

8. 在“图片属性”对话框中可以进行哪些属性的设置？怎样将图片设置成文字环绕的形式？

9. 在“表格属性”和“单元格属性”中，cellspacing 和 cellpading 分别代表什么意思？

10. 如何设置单元格内容和单元格上、下、左、右边框的距离？

11. 在 FrontPage 中，针对 css 回答下列问题：

(1) 如何定义一个 css 文件？

(2) 在一个网页中，如何引用用户定义的 css 文档？

(3) 如果网页中有一个 table，要设置<table>标记的 class 属性，如何操作？

12. 在 html 中，一个表单的<form></form>标记对将产生换行，如果使用表格(table)进行页面布局，如何手工调整<form></form>标记对的位置？

二、综合设计题

利用 FrontPage、Photoshop 或其他媒体制作和页面制作工具，结合 HTML 代码的手工调整，设计并制作个人主页，具体要求如下：

(1) 个人主页的内容要全面反映个人相关信息。

(2) 页面布局、图片、文本等元素要美观大方。

(3) 代码简洁。

(4) 灵活使用 CSS，便于代码维护。

(5) 所用素材和页面文件及文件夹命名规范，各种链接使用相对路径。

第5章 客户端编程

【本章导读】

网页不仅仅是由HTML标记构成的，通常还包含脚本程序，它们增加了页面的交互和计算能力，使网页更生动、功能更强大。因此，制作网页除了必须掌握HTML的各种标记外，还应该熟悉脚本语言编程。脚本语言编程包括两个方面，一方面是客户端编程，另一方面是Web服务器端编程，它们分别在Web浏览器中和Web服务器上执行。

本章介绍客户端的编程问题，首先讲解Web浏览器和客户端脚本程序的关系，对浏览器的工作原理进行分析，它是理解脚本程序和脚本编程的关键；然后以JavaScript语言为例，讲解客户端脚本编程问题，包括JavaScript程序设计语言基础、事件驱动及事件处理、对象及其操作、常用内部对象及函数以及HTML文档对象模型(DOM)，根据在Web开发中的具体需求，详细讲解Web开发中有关表单编程中遇到的问题，包括数据的获取、可靠性验证、网页安全以及网页间的参数传递，并提供大量的实用代码；最后给出多个综合案例。

【知识要点】

5.1节：计算机程序、程序设计语言、源程序、解释执行、程序编译、程序运行。

5.2节：浏览器脚本引擎、客户端脚本语言、JavaScript脚本语言、JScript脚本语言、<script>标记、文件包含。

5.3节：JavaScript脚本语言基本符号、数据类型、弱数据类型、变量、运算符、表达式、程序语句、顺序语句、分支语句、重复语句、函数。

5.4节：类、对象、对象操作、new操作。

5.5节：JavaScript内部对象、String对象、RegExp正则表达式对象、Math对象、Date对象、Array数组对象、JavaScript预定义函数。

5.6节：浏览器对象模型(BOM)、window对象、location对象、history对象、screen对象、navigator对象。

5.7节：HTML文档对象模型(DOM)、document对象、body对象、HTML元素内存对象。

5.8节：AJAX技术、客户端和服务器的异步通信、页面局部刷新。

5.9节：折叠式菜单、树形菜单、数据有效性验证、网页安全。

5.1 计算机程序与程序设计语言

人们使用计算机，确切地讲，是使用计算机软件。计算机软件即计算机程序。在信息社会，程序不应该是一个深奥的专业术语，随着软件集成开发环境的发展，编程也不再是专业

人员的专利,程序应该成为信息社会人们基本信息素养的一部分。

5.1.1 程序设计语言

计算机程序是用户为了达到某种目的而编写的可以控制计算机运行的一组指令序列。程序设计语言(Programming Language)是用于编写计算机程序的语言。每一种程序设计语言都包括语法和语义两个方面。语法表示程序的结构或形式,即表示构成程序的基本符号、符号之间的组合规则。语义表示程序的含义,表示按照语法规则所表示的各种书写内容的含义。

程序设计语言很多,但程序设计语言的基本成分都是相似的,主要包括:①基本符号,定义语言所使用的字符集、保留字、标识符、注释等;②数据和数据类型,对程序所处理的数据的描述;③常量和变量,声明变量;④表达式和运算符,数据处理规则;⑤基本语句,分为顺序语句、分支语句和重复语句,表达业务逻辑;⑥函数,实现模块化和结构化编程。

对于程序设计语言,可以从不同的方面进行分类。按照语言级别可以分为低级语言和高级语言。①低级语言与特定的机器有关,功效高,但使用复杂。低级语言有机器语言和汇编语言。机器语言是基于机器基本指令集的,或者是操作码经过符号化的基本指令集;汇编语言是机器语言中地址部分符号化的结果。②高级语言是一种更加符号化的语言,接近于自然语言,和机器基本上没有紧密关系。

现在,更多的是按编程思想分为过程式程序设计语言和面向对象的程序设计语言。过程式程序设计流行于 20 世纪 90 年代以前,自顶向下逐步求精的结构化程序设计是软件开发的主要方法,直到现在,这种结构化的程序设计思想仍然被广泛地采用。Pascal、C、BASIC、FORTRAN 等高级语言很好地实现了结构化编程的思想,通过过程和函数(又称子程序),把一个复杂的问题划分成几个相对简单的子问题,如果子问题还比较复杂,再继续划分,最后将划分后的每个小问题用过程和函数来实现。

20 世纪 90 年代兴起的面向对象技术对人们近半个世纪来的软件开发思想产生了深刻的变革。这一技术强调利用软件对象进行软件开发,它将自然界中的物理对象和软件对象相对应,建立了类和对象的概念。在面向对象技术中,不仅用对象实现了数据和操作的封装,还通过消息映射在事件和函数之间进行关联,键盘、鼠标等事件的发生会发出消息,消息来激活函数,函数之间的联系不再是显式的调用,这样就降低了函数的耦合度。对于复杂系统面向对象技术可以提高系统的可扩充性和代码重用的层次。当前流行的 C++、Java 都是典型的面向对象程序设计语言。

5.1.2 程序开发及其运行

用计算机程序设计语言编写的程序称为源程序。程序的执行分为解释执行和编译后执行两大类。解释执行就是在某个环境下逐条执行程序语句,当遇到程序错误时,则可能停止运行。目前,网页中的客户端脚本程序就是由 Web 浏览器中的脚本引擎解释执行的。更多的计算机程序采用了源程序、编译、连接到运行的这样一个简单的开发过程,最终形成一个可在计算机操作系统上运行的可执行文件。这个可执行文件被安装在计算机上,它是早期的程序主要应用模式,同时也是 20 世纪 80 年代 C/S 计算模式下的主要应用形式。程序开发过程如图 5-1 所示。

编程人员利用开发工具(如 Visual C++、MyEclipse 等)来编写的程序(即源程序),然后对源程序进行编译、连接操作,最终形成一个可执行的程序,即一个 exe 文件。这种开发模式一直影响到 C/S 计算模式,直到 Web 的出现,一种基于 Web 服务的程序开发和运行模式开始出现并被广泛地应用,基于 Web 的软件开发开始成为程序设计和应用的主流。

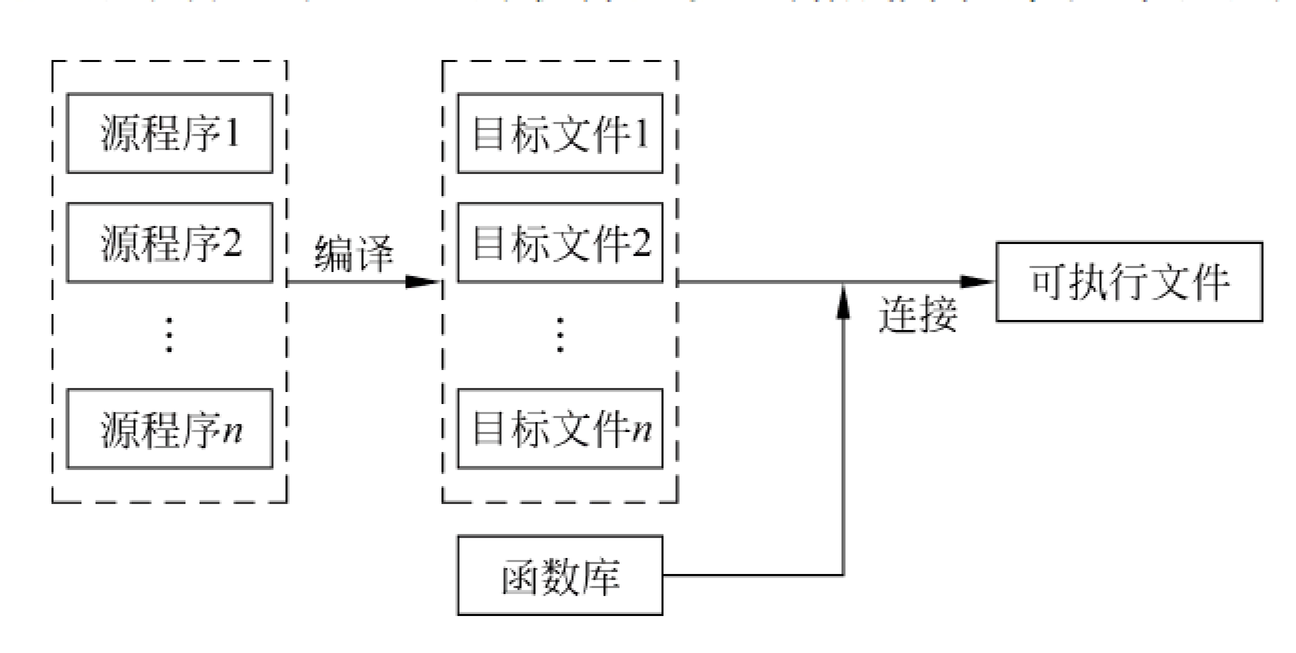


图 5-1 早期的编译型计算机程序开发过程

5.2 浏览器与客户端脚本程序

客户端脚本程序是指在客户的浏览器中运行的程序,这种程序不需要事先编译。如果浏览器从服务器上下载的网页中包含客户端脚本程序,浏览器将对脚本程序代码进行解释执行,即浏览器通过自带的脚本引擎对 HTML 文档中的脚本程序进行分析、识别、解释并执行。

5.2.1 浏览器与客户端脚本引擎

浏览器之所以能够解释执行网页中的客户端脚本程序,是因为浏览器中内置了脚本引擎模块,可以解释执行网页中的客户端脚本程序。客户端脚本程序通常是用脚本程序语言书写的,脚本程序语言和传统的编译型程序设计语言(如 C/C++、Java 等)相比,在语法结构上类似,最大的不同是脚本程序不需要编译、连接过程,即不生成在操作系统下运行的 exe 文件,而是直接在浏览器中,被浏览器解释执行。在解释执行过程中,如果程序存在错误,浏览器将停止程序的执行,并在浏览器窗口的状态栏中显示"网页存在错误"的提示。

由于安全方面的原因,可以使浏览器禁止脚本程序的运行。例如,在浏览器的"Internet 选项"对话框中,选择"安全"选项卡,选择 Internet,单击"自定义级别",在安全设置列表中,可以在"活动脚本"中选择"禁用"、"启用"或"提示"。如果选择"禁用",则浏览器将不执行网页中的客户端脚本程序。

5.2.2 脚本语言规范与主要的客户端脚本语言

1995 年年末,网景公司(Netscape)[①]发布了其客户浏览器脚本程序语言 JavaScript。随

① 美国著名的浏览器厂商,1994 年成立,同年 12 月,推出第一款商用浏览器 Navigator,创始人之一就是马克·安德森(Marc Andreessen)。在微软公司进军浏览器市场以前,其市场份额一度达到 90%,1998 年 11 月 24 日,AOL 公司收购网景公司。

后，Netscape 将 JavaScript 语言规范提交给欧洲计算机制造协会（European Computer Manufacturing Association，ECMA）进行审定。1997 年 6 月，ECMA 发布了名为 ECMAScript Edition 1 的脚本语言规范，即 ECMA-262。ECMAScript 不与任何具体浏览器相绑定，ECMAScript 描述了脚本程序语言所具有的通用属性，包括语法、类型、语句、关键字、保留字、运算符、对象等，这些内容需要由具体的浏览器实现。

1998 年，微软公司在 IE 4.0 中发布了 JScript 3.0，宣称成为第一个遵循 ECMA 规范来实现的 JavaScript 脚本引擎。一年后，Netscape 发布其符合 ECMA 规范的 JavaScript 1.3。1999 年 12 月，ECMAScript 第三版发布，该版本成为脚本程序语言所遵循的标准，目前的所有脚本语言都可以看做 ECMAScript 规范的一种具体实现。例如 JavaScript 就是由三个部分组成的，即 ECMAScript 规范、文档对象模型（DOM）和浏览器对象模型（BOM）。

主要的客户端脚本语言有 JavaScript、JScript 和 VBScript 等，其中，JavaScript 是最早的客户端脚本语言，它是由 Netscape 随着其 Navigator 浏览器一起推出的，是浏览器默认的脚本程序语言。后来微软公司开发了 IE 浏览器，为了在 IE 中支持 JavaScrip 脚本程序，微软公司开发了一个类似的脚本语言，即 JScript。可以说，在 IE 浏览器中，JavaScrip 和 JScript 是等价的，两者在语言上几乎没有什么区别，但是两者提供的对象差别较大。

根据 HTML 规范，在网页中书写脚本程序，脚本程序应该书写在<script>…</script>标记对内。在网页中包含脚本程序的一般形式是：

```
<script type="value" language="value">
    语句部分
</script>
```

script 有两个重要属性。①type 属性，规定脚本的 MIME 类型。MIME 类型由两部分组成，即媒介类型和子类型，对于 JavaScript 来讲，其 MIME 类型为 text/JavaScript。type 属性标识了<script>…</script>之间的内容类型。②language 属性，用于设定脚本程序语言，取值可以是 JavaScript、JScript、VBScript 等，如果是 JavaScript，可以省略不写。另外，在 language 参数中，还可以指定脚本语言的版本号，例如：language="Javascript 1.3"。

和传统的程序设计一样，在 JavaScript 编程中，也可以将一些公用的函数保存为独立的文件，然后在其他的网页中，在<head></head>中，把其他 JavaScript 文件包含进来，一般形式是：

```
<script src="include-filen-ame.js"></script>
```

脚本程序可以出现在网页的头部，也可以出现在网页的文档体中，其中出现在文档头部的脚本程序通常是一些函数，这些函数只有在显式地被调用时才被执行。在介绍具体的客户端脚本程序设计以前，先看两个包含 JavaScript 客户端脚本程序的简单网页示例。

【例 5-1】 编写一段脚本程序，检测浏览器中对 JavaScript 脚本程序的支持情况。

分析：不同的浏览器对脚本程序的支持情况不同，下述代码可以检测浏览器对 JavaScript 脚本程序不同版本的支持情况。

代码清单：exa5-1.htm

```
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<h1>JavaScript 检测</h1><hr>
<!-- JavaScript 支持性检测 -->
<script type="text/javascript">
    document.write("浏览器支持 JavaScript!<br><br>");
</script>
<!-- JavaScript 版本检测 -->
<script type="text/javascript1.0">
        document.write("浏览器支持 JavaScript 1.0<br>");
</script>
<script type="text/javascript1.1">
    document.write("浏览器支持 JavaScript 1.1<br>");
</script>
…
<script type="text/javascript1.5">
    document.write("浏览器支持 JavaScript 1.5<br>");
</script>
</body>
</html>
```

分别使用 IE 6.0、Maxthon2(遨游)和 Mozilla Firefox 2.0 打开上述网页文件 exa5-1.htm，在 IE 6.0 和 Maxthon2(遨游)浏览器中，显示浏览器支持 JavaScript 1.1、1.2 和 1.3，因为 Maxthon 浏览器采用了 IE 内核，因此，两者对于 JavaScript 版本的支持一致。在 Mozilla Firefox 2.0 浏览器中，显示浏览器支持 JavaScript 1.0 到 JavaScript 1.5 的全部 JavaScript 版本。

和所有的软件开发一样，Web 页面中的程序也需要对程序进行结构化设计。在 JavaScript 中，同样提供了函数定义与函数调用功能，以支持结构化程序设计。由于浏览器浏览的 Web 页是顺序地从 Web 服务器调出，并由浏览器解释执行的，函数必须遵循先定义(一般放在<head>…</head>内)后调用(在<body>…</body>内)的原则。

【例 5-2】 编写一个 JavaScript 函数，求一个正整数 n 的阶乘。

分析：模块化编程同样适用于脚本程序，函数可以写在 HTML 页面的任何位置，从良好的编程习惯上讲，函数定义通常写在<head></head>中。

代码清单：exa5-2.htm

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<script type="text/javascript">
function fact(n)
{
if (n==0)
    return 1;
else
    return n*fact(n-1);
}
```

```
</script>
</head>
<body>
<p>fact(5) =
<script type="text/javascript">
    document.write(fact(5));
</script>
</body>
</html>
```

在浏览器中打开上述网页文件，在浏览器中显示：

```
fact(5) = 120
```

JavaScript 程序和一般的编译型程序不同，它没有一个操作系统调用的主程序(如 C 语言中的 main()函数)，一个 JavaScript 函数定义时并不发生作用，只有在引用时(函数定义后的 document.write 语句)才被激活。

因为不同的浏览器和浏览器版本支持的脚本语言版本不同，因此，在书写客户端脚本程序时，应该根据浏览器的种类和版本，使用合适的内置对象和浏览器对象，否则，如果程序中使用了一些高版本脚本语言包含的对象，则网页在不支持该版本的浏览器中，或低版本的浏览器中可能不能正常地显示。

5.3　JavaScript 程序设计基础

JavaScript 是一种基于对象、事件驱动的浏览器脚本编程语言。1995 年，Netscape 开始研发一种开发代号为“魔卡(Mocha)”的脚本语言，根据 Netscape 的 Live 战略，该语言很快更名为 LiveScript，到了 1995 年年末，为了迎合市场对 Java 语言的热情和市场营销的需要，该脚本语言正式地命名为 JavaScript。由于名称的缘故，容易使人们将 JavaScript 和 Java 联系在一起，但是，无论从思想和应用环境，JavaScript 和 Java 语言都有着本质的区别。

5.3.1　JavaScript 基本符号

任何一种程序设计语言都有其自身的字符集和基本符号，它们按照语法构成程序语句，然后语句再构成程序。

1. 基本字符

JavaScript 语言的基本字符有字母(a、b、…、z，A、B、…、Z)、数字(0、1、…、9)和特殊符号(＋、－、* 、/、<、＝、>等)三大类。

同 C 语言一样，JavaScript 中同样以反斜杠(\)开头来表示不可显示的特殊字符，通常称为控制字符。例如，'\10'表示换行(\n)，'\13'表示回车(\r)。

2. 关键字

关键字又称为保留字，它是由字母构成的具有固定含义的单词，如 var 代表变量说明，if

表示条件语句等。JavaScript 的关键字很多,可以参考 JavaScript 的专门书籍。

需要特别注意的是,JavaScript 是识别大小写的,在 JavaScript 中,关键字需要小写,如果书写有误,将显示“JavaScript 脚本错误”的警告。

3. 标识符

标识符是指表示常量、变量和函数等名称的符号。标识符分为标准标识符和用户自定义标识符。标准标识符是表示标准常量和标准函数等名称的符号。JavaScript 中的标准常量和标准函数见下面的介绍。用户自定义标识符是指用于说明常量、变量和函数等名称的符号。用户自定义标识符必须以字母开始,由任意的字母、数字和“_”组成。用户在命名标识符时应该有一定的命名规范。第一,用户自定义标识符不应该与标准标识符重名。第二,用户自定义标识符要尽可能反映它所代表的对象的含义。如:用 Name 代表名称比用 x 代表名称可读性更强。第三,如果是一个变量名,还应该尽可能反映变量的数据类型和作用域,如用 nUserID 表示一个用户标识,最前面的小写字母 n 代表变量为整数类型。第四,大小写混写和较长的标识符可以把含义表达得更清晰。总之,好的命名规范可以提高程序的可读性,便于程序的维护。

4. 注释

为了增加程序的可读性,一般在程序中增加注释语句,注释内容没有语法要求,其内容仅仅起提示作用。在 JavaScript 中,注释以“//”字符引导,注释可以单独一行,也可以在语句行的后面。

注释一般分为序言性注释和描述性注释两种类型。序言性注释出现在模块的首部,其内容一般包括有关模块功能的说明、界面描述、调用语句格式、所有参数的解释和该模块需调用的其他模块名等。还包括一些重要变量的使用、限制及其他信息。描述性注释嵌在程序之中。描述性注释又有功能性的和状态性的,功能性注释说明程序段的功能,通常可放在程序段之前;状态性注释说明数据的状态,通常可放在程序段之后。

需要说明的是,与 HTML 的注释(<!--…-->)不同,当在服务器端处理脚本时,JavaScript 中注释将被删除,而不是被送到浏览器。若需要客户端浏览器看到脚本中的注释,应该使用 HTML 注释,将注释加进 HTML 页。此时,注释将返回给浏览器。

5.3.2　数据和数据类型

在程序设计中,数据类型就是定义一段计算机内存空间的解析规则,说明变量就是要说明变量名称及其数据类型。数据类型决定了变量所占内存空间的大小,同时也决定了变量的取值范围。对于程序中声明的全局变量,程序在运行时,操作系统会为其分配空间。直到程序运行结束,所占用的内存空间被释放。

JavaScript 脚本程序语言是一种解释性的程序语言,所写的脚本程序不需要编译和链接,它采用边读边执行的方式运行。在数据类型方面,JavaScript 提供了三种基本的数据类型用来处理数字和文本。JavaScript 提供的数据类型有数值(整数和实数)、字符串型(用西文双引号“"”或西文单引号“'”括起来的字符)、布尔型(true 和 false,均为小写)。

JavaScript 采用弱类型的形式,没有预定义的类型名,变量不必事先声明其数据类型,

可以在使用或赋值时确定其数据类型。此外,一个变量的类型在使用时还可以被改变。通常情况下,可以先给变量赋一个初值,通过初值的类型来声明该变量的数据类型,这更加符合一般的程序设计思想,例如:

```
var x = "hello";
```

上述语句声明了一个变量 x,通过为 x 赋初值"hello",将 x 作为一个字符串类型变量。在后面的语句中,可以给 x 赋一个整数,如 x=100,这样变量 x 就成为整数类型了。在 JavaScript 中,上述变量类型的改变不会和其他程序设计语言一样出现赋值不相容的错误。

不同的数据类型之间可以进行转换,例如:将字符串转换为相应的整数,将整数转换为字符串数据等。这些转换被封装在相应的类或对象中,字符串到数值的转换函数可见 5.5.1 节的 String 对象。如果是数值型转换为字符串,最简单的方法就是将一个字符串和数值做"+"运算,结果即为字符串类型。例如:

```
var age = 18;
var agestr = age * 2 + "";
```

上述代码,第一个语句设置 age 为整数,第二个语句 agestr 则为字符串。通过整数和一个字符串,例如空串相加,即可得到字符串。

5.3.3 常量和变量

1. 常量和常量定义

常量是指在程序运行过程中,其值不发生变化的量。常量有字面常量和符号常量两种。字面常量就是一些数值或字符串,例如 3.14、"hello"等都是字面常量。根据数据类型的不同,常量可分为整数常量、实数常量、字符常量等。字符型常量是指使用单引号(')或双引号(")括起来的一个或多个字符。如'a'、"hello"、"5123"、"x+y" 等,其中单引号括起来的为单个字符,双引号括起来的为字符串。符号常量是指为一个常量起一个名字,即常量名,常量名是一个用户自定义标识符。例如:用 pi 代表圆周率 3.14。

常量命名有两方面的好处,首先恰当的常量名称可以增加程序的可读性;其次,使用常量名便于程序的维护。例如,可以命名常量 pi=3.14,如果希望提高求解精度,可以修改 pi 的定义为 pi=3.1416,这种修改只在一处进行,不会产生不一致的情况,而且修改简单。

2. 变量和变量说明

变量是指在程序执行过程中其值发生变化的量。每一个变量都有一个变量名,对应一个特定的内存空间。变量有两个重要的属性,一个是数据类型,一个为操作运算。数据类型决定了变量所占内存空间的大小,也决定了数据的取值范围和操作运算。

在 JavaScript 中,变量命名的一般形式是:

```
var <变量名表>;
```

其中,var 是 JavaScript 的保留字,表明接下来是变量说明,变量名表是用户自定义标识符,变量之间用逗号分开。和 C/C++等程序设计语言不同,在 JavaScript 中,变量说明不需

要给出变量的数据类型。此外,变量也可以不说明而直接使用。例如:

```
var x,y;
```

定义了两个变量,名称分别为 x 和 y,没有给出具体的数据类型,也没有赋予其值。

再如:

```
var myName = "John"
```

定义了一个变量 myName,同时赋予了它一个字符串值。在 JavaScript 中,变量可以不作声明,在使用时,数据的类型将确定变量的类型。

3. 变量的作用域

变量的作用域由声明变量的位置决定,决定哪些脚本命令可访问该变量。在函数外部声明的变量称为全局变量,其值能被所在 HTML 文件中的任何脚本命令访问和修改。在函数内部声明的变量称为局部变量。只有当函数被执行时,变量被分配临时空间,函数结束后,变量所占据的空间被释放。局部变量只能被函数内部的语句访问,只对该函数是可见的,而在函数外部则是不可见的。

5.3.4 表达式和运算符

表达式是指将常量、变量、函数、运算符和括号连接而成的式子。根据运算结果的不同,表达式可分为算术表达式(结果为整数或实数)、字符表达式(结果为字符或字符串)和逻辑表达式(结果为 true 或 false)。

JavaScript 提供了丰富的运算功能,包括算术运算、关系运算、逻辑运算和连接运算等。

1. 算术运算符

JavaScript 中的算术运算符有单目运算符和双目运算符。双目运算符包括+(加)、-(减)、*(乘)、/(除)、%(取模)、|(按位或)、&(按位与)、<<(左移)、>>(右移)等。单目运算符有-(取反)、~(取补)、++(递加 1)、--(递减 1)等。

2. 关系运算符

关系运算又称比较运算,关系运算符包括<(小于)、<=(小于等于)、>(大于)、>=(大于等于)、=(等于)和!=(不等于)。

关系运算的运算结果为布尔值,如果条件成立,则结果为 true,否则为 false。

3. 逻辑运算符

逻辑运算符有 &(逻辑与)、|(逻辑或)、!(取反,逻辑非)、^(逻辑异或)。

4. 连接运算符

连接运算用于字符串操作,连接运算符有+(用于强制连接),将两个或多个字符串连接为一个字符串。

5. 三目操作符

三目操作符“?:”格式为：

```
操作数?表达式 1: 表达式 2
```

三目操作符“?:”构成的表达式，其逻辑功能为：若操作数的结果为 true，则表述式的结果为表达式 1，否则为表达式 2。例如 max=(a>b)? a:b; 该语句的功能就是将 a，b 中的较大的数赋给 max。

5.3.5　基本语句

所有的程序设计语言，无论是过程式程序设计语言还是面向对象的程序设计语言，其程序设计语句都可以分为三种类型，即顺序语句、分支语句和重复语句，使用这三种语句就可以描述用户的所有业务逻辑。

1. 顺序语句

从本质上讲，在一个程序中，语句总是从上而下顺序执行的。在过程式的程序设计语言中，这种顺序是程序本身所显式地定义的。例如 C 语言中的函数调用，遇到函数调用，转去执行相应的函数，执行结束后返回。在面向对象的程序设计语言中，函数的调用变得更加复杂，它是在程序的运行过程中，由事件触发消息，由消息来激活函数。这样的事件驱动机制，使函数的调用变得不再像过程式程序设计中的显式调用那么清晰，但这种消息映射(message map)机制降低了函数之间的耦合度，大大增强了软件系统的可维护性。在函数内部，语句仍然是从上到下顺序执行的。

无论是 C、C++、Java，还是 JavaScript，由于语句语法的需要，有时候需要将多个语句看做逻辑上的一个语句，此时需要将这多个语句用一对花括号“{”和“}”括起来，语句之间用分号分开，称为语句块。在语句块内部，语句从上而下顺序执行。

2. 分支语句

在 JavaScript 中，实现分支结构的语句有三种，它们是条件判断语句 if 语句、if…else…语句和开关语句 switch 语句。

1) if 语句

if 语句的一般形式是：

```
if  (<条件表达式> )
<语句>;
```

if 语句首先计算条件表达式的值，若计算结果为 true，或非 0 值的数，包括正数或负数，则执行语句部分，否则执行 if 语句下面的语句。if 语句逻辑功能如图 5-2 所示。

语句部分逻辑上是一个语句，如果语句部分需要多个语句来实现，应将这多个语句用“{”和“}”括起来，形成语句块，作为逻辑上的一个语句。例如：

```
if (a>=b)
```

```
{
    max = a;
    min = b;
}
```

2）if…else…语句

if…else…语句先计算条件表达式的值，根据计算结果确定要运行的语句。一般形式是：

```
if  (<条件表达式> )
  <语句 1 >;
else
  <语句 2 >;
```

if…else…语句首先计算条件表达式的值，若为 true，或非 0 值的数，则执行语句 1，否则，执行语句 2。逻辑功能如图 5-3 所示。

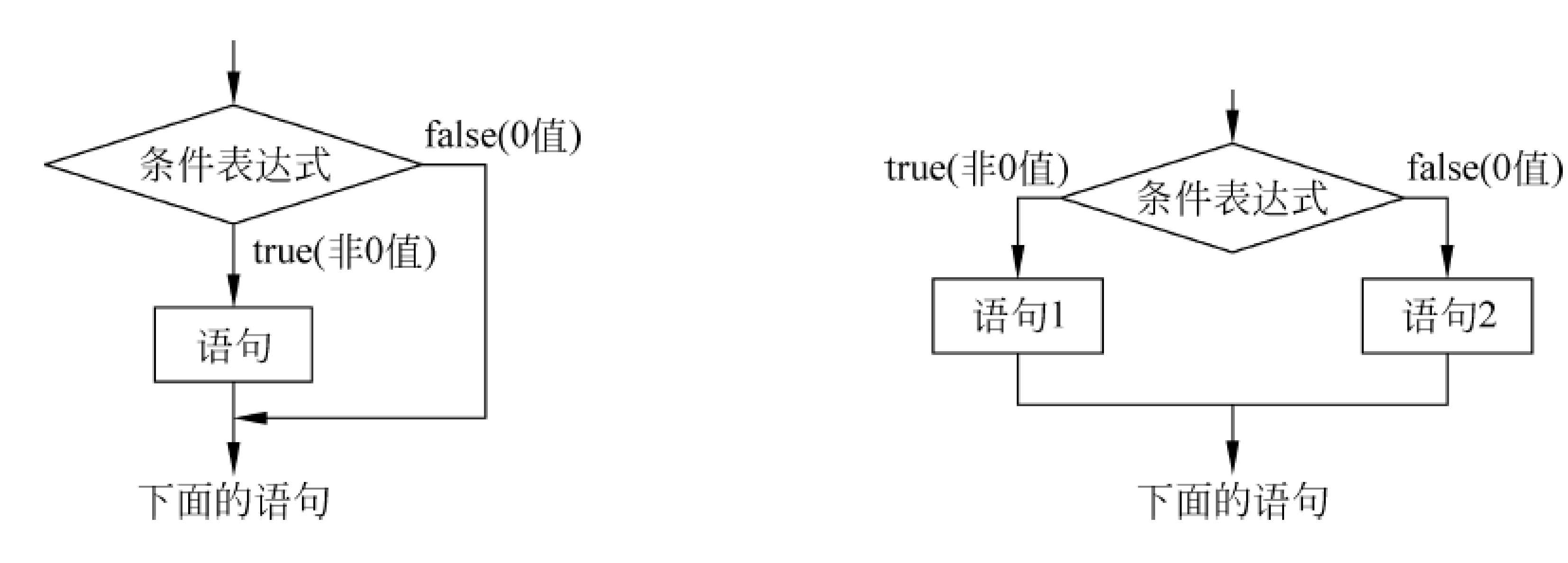


图 5-2 if 语句逻辑功能　　图 5-3 if…else…语句逻辑功能

在存在两种分支的情况下，if…else…语句可以很好地描述这种逻辑。在多种分支的情况下，可以使用 if…else…语句的嵌套进行描述，但是嵌套降低了程序的可读性。为此，对于多种分支的情况，JavaScript 提供了 switch 语句。

3）switch 语句

switch 语句提供了 if…else…多层嵌套结构的一种变通形式，可以从多个语句块中选择执行其中的一个。switch 语句提供的功能与 if…else…语句类似，但是可以使代码更加简练易读。

switch 语句的一般形式为：

```
switch (<数值或字符串表达式>)
{
    case 表达式 1
        语句块 1;
        [break;]
    case 表达式 2
        语句块 2;
        [break;]
        ⋮
    case 表达式 n
        语句块 n;
        [break;]
```

```
    [default:
      语句块 n+1;
    ]
}
```

在 switch 语句的开始是一个条件表达式，该条件表达式可以是一个整数、实数或字符串表达式。表达式的结果将与结构中每个 case 的表达式值比较。如果匹配，则执行与该 case 关联的语句块。在语句块中，如果包含 break 语句，则退出 switch 语句，否则，将继续下面的 case 匹配。switch 语句的逻辑功能如图 5-4 所示。其中虚线或框为可选语句及流程。

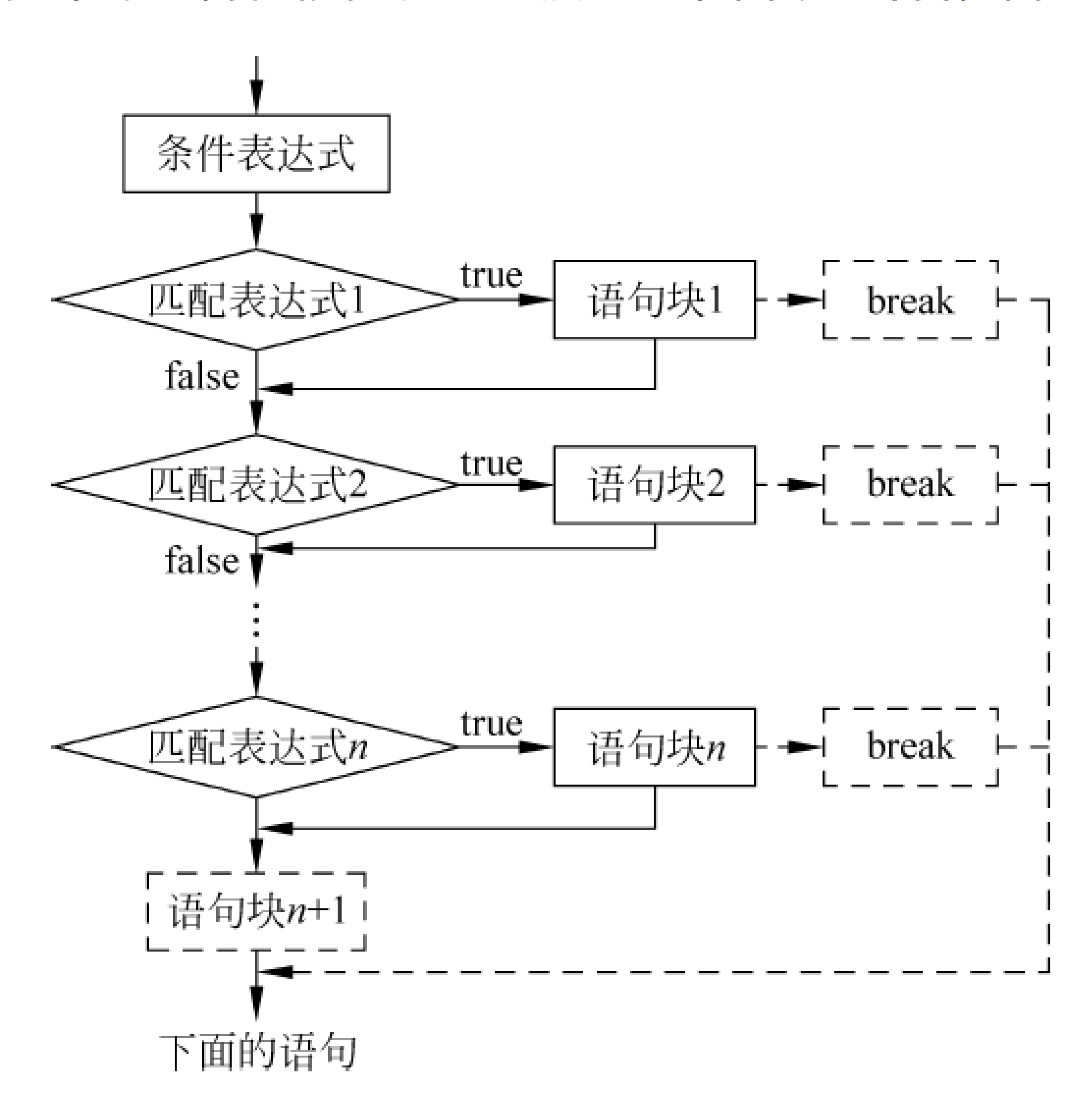


图 5-4　switch 语句逻辑功能

3. 重复语句

当部分语句需要反复执行时，需要使用重复语句。重复语句总是由循环体和循环终止条件两部分组成。在 JavaScript 中可使用下列两种形式的重复语句。

1）while 循环语句

while 循环语句是先判断循环终止条件，然后再执行循环体，因此循环体可能一次也不被执行。while 循环语句的一般形式是：

```
while (<条件表达式>)
    <语句>;
```

首先计算条件表达式的值，若计算结果为 true，或非 0 的数，则执行循环体语句，然后无条件地返回 while 语句的开始位置，继续计算条件表达式的值。如果条件表达式计算的结果为 false，则结束循环，执行 while 循环语句下面的语句。逻辑功能如图 5-5 所示。

2）for 循环语句

一般形式为：

```
for (表达式 1; 表达式 2; 表达式 3)
```

<语句>;

执行过程如下。

(1) 计算表达式 1。

(2) 计算表达式 2,如果结果为 true,或非 0 值,则执行循环体,然后转 Step3。否则,结束循环。

(3) 计算表达式 3。

(4) 无条件转步骤(2)。

(5) 循环结束,执行 for 语句下面的语句。

逻辑功能如图 5-6 所示。

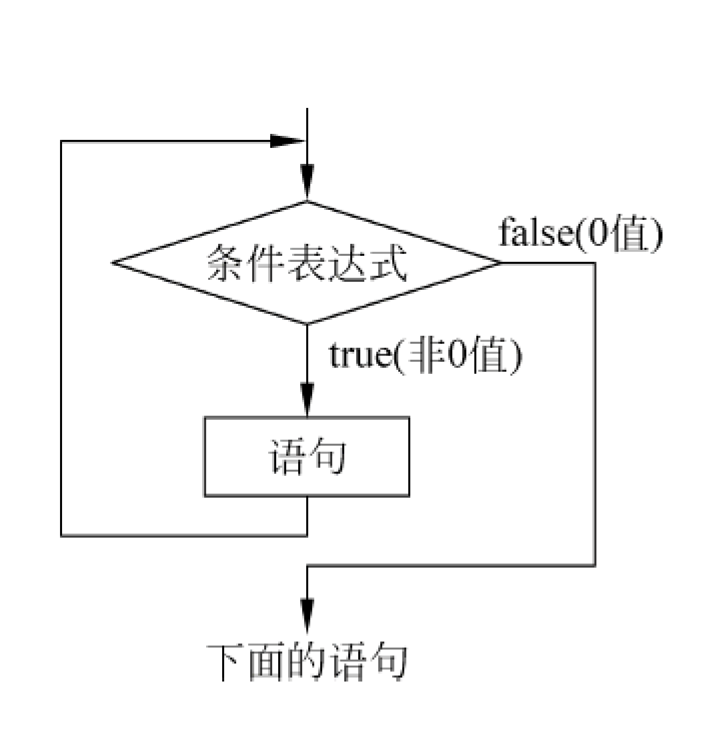


图 5-5 while 语句逻辑功能

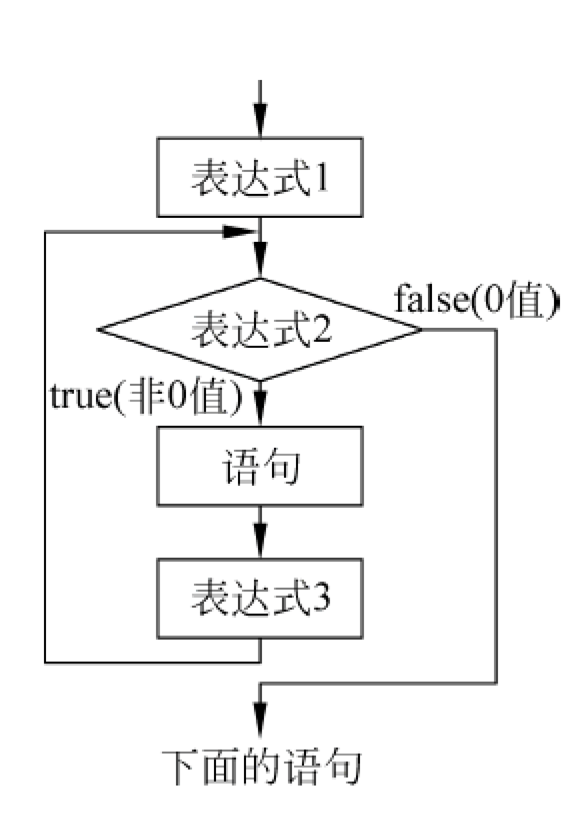


图 5-6 for 语句逻辑功能

在 for 语句中,表达式 1、表达式 2 和表达式 3 可以省略其中的一个或多个,但它们之间的分号不能省略。例如,可以写成下面的 for 语句形式:

```
for (; ;)
{
…
}
```

上述 for 语句是一个死循环,在循环体中必须包含 break 语句,以结束循环的执行。另外,循环体的内部也可以包含循环语句,构成多重循环,即循环的嵌套。需要特别注意关键字要小写,例如 for 不应该写成 For。

4. break 和 continue 语句

与 C++ 语言相同,使用 break 语句使得程序执行从 for 语句或 while 语句中跳出,continue 使得跳过循环内剩余的语句而进入下一次循环。

5.3.6 函数

在结构化程序设计中,函数是实现结构化程序设计的主要手段,它把一个系统中需要反复多次执行的部分定义为一个函数。如判断一个数是否为素数、求两个数的最大公约数等。在 JavaScript 语言中,已经给出了许多标准函数,同时也允许用户自己定义函数。

在 JavaScript 中，函数是以 function 开头定义的，由函数头部和函数体构成。一般形式为：

```
function <函数名> (<形式参数表>)
{
    变量说明部分；
    语句(函数体)；
}
```

在函数的定义中，函数名后有形式参数表，这些形式参数变量可能是一个或几个，在 JavaScript 中可通过“函数名.arguments.Length”来检查参数的个数。在函数体中，可以分为变量说明和语句部分，变量说明用于说明该函数所要处理的数据，语句部分是对变量的处理。在 JavaScript 中，由于采用弱数据类型，变量说明部分可以省略不写。在函数体的语句部分必须包含一个返回函数值的语句，即 return 语句。

5.4　类与对象

在软件系统的开发中，面向对象的程序设计是目前最主流的程序设计思想，大部分的软件开发工具也是面向对象的，例如 C++、Java 等。面向对象的思想将问题域中的每一个实体都映射为软件系统中的一个对象，通过类和对象，实现了数据和数据操作的封装，极大地增强了系统的灵活性、可维护性和可扩展性。

5.4.1　类与对象的概念

类(Class)和对象(Object)是面向对象程序设计(Object Oriented Programming，OOP)的灵魂。所谓“类”，是指一种包含成员变量和成员函数的数据结构。类和传统的过程式程序设计中的数据类型相似，它不占用内存空间，主要目的是用于声明对象。对象是用类来声明的数据结构，如果将类比作数据类型，对象就是相应数据类型的变量，在内存中分配特定的空间、存储数据，类决定了对该空间的语义解析。

JavaScript 并不是一个完整的面向对象的程序设计语言，它不提供关于对象的抽象、封装、继承、派生、多态等功能。但它可以使用浏览器的内置对象，也允许用户自定义对象，从而分享面向对象程序设计带来的好处。

因为 JavaScript 是一种基于对象的程序设计语言，理论上讲，JavaScript 并没有定义类的功能。在 JavaScript 中，所谓定义“类”，其实是一种模拟的自定义类型。有两种常见的形式，介绍如下。

1. 利用函数构造类

利用函数构造类的方式和“函数”定义的语法相同，而且这样的函数成为该类的构造函数。用户用函数定义来定义类，然后用 new 语句创建该类的一个实例(对象)。一般形式是：

```
function className(< prop1,prop2, …>)
```

```
{
    this.prop1 = prop1;                    // 属性
    this.prop2 = prop2;
    …
    this.method1 = FunctionName1;          // 方法,需要定义对应的函数
    this.method2 = FunctionName2;
    …
}
```

上述定义即定义了一个名为 className 的类,它包含的属性为 prop1,prop2,…,包含的成员函数是 method1,method2,…。

2. 利用原型 prototype 定义类

在 JavaScript 中,类是以函数的形式来定义的。每个类对象都有一个 prototype 属性对象。用户可以向对象的 prototype 属性添加属性和方法。

例如,为 JavaScript 内置对象添加方法:

```
<script lanuage = "javascript">
Array.prototype.max = function()
{
    var i, max = this[0];
    for (i = 1; i < this.length; i++)
    {
        if (max < this[i])
            max = this[i];
    }
    return max;
}
</script>
```

删除 String 对象两侧的空格,代码如下:

```
String.prototype.trim()
{
    return this.replace(/(^\s+)|(\s+ $ )/g, "");
}
```

除了可以为内置对象添加属性和方法外,可以为用户定义的类添加方法,示例代码如下:

```
function TestObject(name)
{
    this.m_name = name;
}
TestObject.prototype.ShowName = function()
{
    alert(this.m_name);
}
```

上述代码定义了一个类 TestObject,包含一个属性 m_name,一个方法 ShowName()。

【例 5-3】 定义一个 JavaScript 类，包含一个成员变量、三个成员函数，并用不同的方法定义成员函数，比较它们的不同。

分析：在 JavaScript 中，可以使用 function 定义类。对类成员函数可以用不同的方法定义，不同方法定义的函数不同。

代码清单：

```
<html>
<head>
<script language = "javascript">
function f1()
{
  return this.m_name;
}
//////////////////////////////////////////////////////////////////////////////
//定义 TestObject 类
function TestObject(name)
{
  this.m_name  =  name;
  //成员函数定义方法 1
  this.f1 = f1;
  //成员函数定义方法 2
  this.f2 = function()
  {
    document.write("this.f2()<br>");
  }
}
//成员函数定义方法 3
TestObject.prototype.ShowName  =  function()
{
  document.write(this.m_name + "<br>");
}
</script>
<head>
<body>
<script language = "javascript">
a = new TestObject("Obj A");
b = new TestObject("Obj B");

a.ShowName();
b.ShowName();

document.write(a.f1 == b.f1);
document.write(a.f2 == b.f2);
document.write("<br>");
document.write(a.ShowName == b.ShowName);
</script>
</body>
</html>
```

在上述代码中，在页面的头部定义了一个 JavaScript 类 TestObject，用三种方法定义了

三个成员函数 f1()、f2()和 ShowName()。在页面的<body>体内，声明了两个对象 a 和 b。在浏览器中打开该页面，显示结果如下：

```
Obj A
Obj B
true
false
true
```

从上述输出结果可以看出，三种方法定义的成员函数不同，如果在类的定义中嵌入函数定义(方法 2)，不同的实例对象将拥有独立的成员函数副本。使用 prototype 定义的成员函数(方法 3)，所有对象拥有同一个成员函数。

此外，不同方式定义的成员函数，JavaScript 解释引擎搜索的顺序也不相同，具体过程是：JavaScript 解释引擎在处理"."或"[keyName]"引用的对象的属性和方法时，先在对象本身的实例(this)中查找，如果找到就返回或执行。如果没有找到，就查找对象的 prototype，查看是否定义了被查找的对象和方法，如果找到就返回或执行，如果没有查找到，就返回 undefined(对于属性)或 runtime error(对于方法)。

5.4.2 对象操作

无论是 C++/Java 等面向对象的程序设计语言，还是 JavaScript 这样基于对象的程序设计语言，定义类的目的是说明对象，对象实现数据的存储，类只是定义了对象的数据结构和内存解析规则。

1. 创建对象

创建对象就是按照类的定义在内存中分配一段空间，并为该空间命名。在 JavaScript 中，使用 new 运算符可以创建一个新的对象。创建对象的一般形式是：

```
myobj = new calssName(参数表);
```

其中，myobj 为创建的对象名称，calssName 是类的名称，参数表用于激活类的相应的构造函数，从而为类中的成员变量赋值。

如创建一个日期对象，代码为：

```
myDate = new Date();
myBirthday = new Date("May 11,1975 6:15:00");
```

这样就创建了两个新的日期对象 myDate 和 myBirthday，myDate 对象取当前的计算机时钟作为其日期值，而第二个 Date 对象 myBirthday 被赋值一个具体的日期。

2. 访问对象属性和方法

对象属性和方法的引用主要使用点(.)运算符，一般形式为：

```
对象名.属性名|方法名
```

或：

```
对象名[属性名|方法名]
```

3. for…in 语句

格式如下：

```
for(对象属性名 in 已知对象名)
```

顺序输出对象各个属性的值，如果是方法，则输出对应的函数代码。例如：

```
function showData(object)
{
    for(var prop in object)
        document.write(object[prop]);
}
```

使用该函数时，在循环体中，可以不知道对象属性的个数，for 可以自动地将属性取出来，直到最后为止。

4. this 关键字

this 是对当前对象的引用，在 JavaScript 中，对象的引用是多层次的，往往一个对象的引用又需要对另一个对象的引用，而另一个对象有可能又要引用另一个对象，这样有可能造成混乱，为此 JavaScript 提供了一个用于指定当前对象的指针 this。

5.5 JavaScript 内置对象及全局函数

每一种程序设计语言都可以分成程序设计语法和标准库两部分。语法部分通常用于定义一种程序语言的基本语法规范，包括字符集、保留字、数据和数据类型、程序语句等内容；标准库则是指为用户提供的一组常用函数库或类库。对于结构化程序设计语言（如 C 语言），标准库通常是一组标准的函数库，包含大量的标准函数。如果是面向对象的程序设计语言（例如 C++、Java 等），标准库则是一组标准类库，包含大量的标准类。使用标准函数库或类库，可以提高用户的编程效率和保证程序代码质量。

由于 JavaScript 是一种基于对象的脚本程序设计语言，它具有面向对象程序设计语言的特点，但又不完全是面向对象的程序设计。这体现在关于类库的设计方面，例如 JavaScript 不仅提供了一组标准类库，同时还提供了一组常用的对象和全局函数。而在面向对象的程序设计思想中，尽量不使用全局对象变量，所有的函数都应该隶属于一个类。不同的 JavaScript 版本，包含的内部对象和函数也不一样，下面介绍一些常用的内部对象和函数。

5.5.1 String 对象

在 JavaScript 中，每个字符串都是一个 String 对象。使用 String 对象时，不需要像一般自定义对象一样用 new 关键字在内存中创建对象，而是可以直接将一个字符串赋给一个变

量。string 字符串对象封装了 JavaScript 中的字符串以及相关的操作。

字符串对象的生成十分简单,而且是隐式的,不使用 new 关键字,例如:

```
var myStr = "Hello";
```

这样,myStr 就是字符串对象了,一个变量被声明为字符串对象之后,它就拥有了这个对象类的属性和方法,可以和一般对象一样,使用对象的方法,取得对象的属性。

1. 字符串对象的属性

字符串对象的属性只有一个,这就是 length(长度)属性,返回字符串的长度。需要说明的是,如果字符串为英文,那么长度属性的值是字母个数加上特殊符号个数,加上空格数;但是如果字符串是中文,每个中文单字占两个英文字符的长度。例如:

```
<script type = "text/javascipt">
    var myStr = "Hello";
    document.write(myStr.length);
</script>
```

上述代码在浏览器中输出字符串"Hello"的长度为 5。

2. 字符串对象常用方法

字符串对象是使用最为频繁的对象,因此也包含更多的常用对象成员函数。常见的 String 对象成员函数见表 5-1。

表 5-1　String 对象常用成员函数

函数及一般形式	功　　能
charAt(Position)	取字符函数,返回字符串对象中下标为 Position 的字符(Position 为大于等于 0,且小于字符串长度的整数)
substring(Position1,Position2) substring(Position)	第一种格式中,返回下标从 Position1 到 Position2 之前的字符串(不含下标是 Position2 的字符)。第二种格式返回下标为 Position 开始直到字符串结束的字符串
indexOf(subStr) indexOf(subStr,StartPosition)	subStr 是一个待查找的字符或者字符串,可以是常量,也可以是变量。在省略 StartPosition 的情况下,此函数将从字符串的第一个字符开始查找;当 StartPosition 参数存在的情况下,这个函数将从字符串中下标为 StartPosition 的字符开始查找;当 StartPosition 超过字符串的长度时,返回－1。当所希望查找的字符串找不到时,返回－1。当字符串中有两个以上的待查找字符串,则返回被搜寻字符串中位置在最前面的待查字符串的下标位置
lastIndexOf (subStr) lastIndexOf (subStr,StartPosition)	同 indexOf()函数类似,只是查找是从字符串尾部往前查找
match(searchvalue) match(regexp)	在字符串内检索指定的值,或找到一个或多个与正则表达式的匹配。参数 searchvalue 规定要检索的字符串值。参数 regexp 为正则表达式。如果没有找到任何匹配的文本,match() 将返回 null。否则,它将返回一个数组,其中存放了与它找到的匹配文本有关的信息

续表

函数及一般形式	功　　能
replace(str1,str2)	将字符串中的字串 str1 用 str2 替换,同时 replace()支持正则表达式,它可以按照正则表达式的规则匹配字符或字符串
toLowerCase()	将字符串中所有的字母变为小写字母,将变化后的结果返回。但是,原字符串内的大小写不变
toUpperCase()	将字符串中所有的字母变为大写字母,将变化后的结果返回。但是,原字符串内的大小写不变
split(separator, limit)	将字符串分割为字符串数组,并返回此数组。separator 为分隔符,limit 为可选参数、分割的次数,如果无此参数为不限制次数
length()	返回字符串的长度,所谓字符串的长度是指其包含的字符的个数

JavaScript 提供了大量的字符串处理函数,功能非常强大,下面是几个简单的例子。例如,在 HTML 的多行文本框输入中,包含了回车换行符,如果要保存到数据库中,则回车换行无法保存;如果要将数据库内容显示,则没有换行。因此,在保存数据库时可以将文本框中的回车换行替换为 HTML 的
标记。代码如下:

```
//设字符串保存在变量 str 中,删除其中的回车、换行符
str = str.replace(/\r/g,"<br>");
str = tstr.replace(/\n/g,"");
```

上述代码使用了正则表达式,"\r"表示回车字符,"/g"表示全部,即将 str 中全部的"\r"替换为
。第二行则表明将全部的换行"\n"替换为空,即删除。如果要将全部的字符"a"替换为"A",可写为 str=tstr.replace(/a/g,"A");。

字符串的打散也是常用的操作之一。例如:

```
var str1 = "aa;bb;cc";
var str2 = "hello";
s1 = str.split(";");
s2 = str.split("");
```

则得到一个字符数组 s1=["aa","bb","cc"],s2=["h","e","l","l","o"]。

与 split 对应的函数是 Array 对象的 join(delimiter)方法,使用给定的分隔符将一个数组合并为一个字符串。例如:

```
var aa = new Array("jpg","bmp","gif","ico","png");
var str = aa.join("|");
```

则字符串 portableList 为"jpg|bmp|gif|ico|png"。

5.5.2 RegExp 对象

在字符串处理中,经常会用到查找某种特定模式的字符串,或者是验证某个字符串是否符合特定的格式。例如,检查一个字符串是否为有效的邮箱地址,检查用户名是否字母和数字的组合等。用来描述这种复杂规则的表达式称为正则表达式(Regular Expression)。在 JavaScript 中,RegExp 对象表示正则表达式,它是对字符串执行模式匹配的强大工具。使

用正则表达式,可以大大减少字符串处理的编程工作量,使数据有效性验证更加准确和高效。

1. 正则表达式的概念

正则表达式的概念最初出现于理论计算机科学的自动控制理论和形式语言理论中,是用于对模型和规则的一种形式描述,20世纪50年代,正则表达式的概念被应用于UNIX的编辑器工具软件,随后被普及开来。许多程序设计语言都支持利用正则表达式进行字符串操作。

正则表达式是一种由普通字符和特殊字符(元字符)组成的字符串匹配模式,在正则表达式中,元字符(Metacharacter)是拥有特殊含义的字符。正则表达式只能使用"/"开头和结束,不能使用双引号,因为双引号是字符串对象的表示方法。

在JavaScript中,正则表达式中可以使用的元字符及其含义见表5-2。

表5-2 JavaScript正则表达式元字符

元字符	含　义	元字符	含　义
.	匹配除换行符以外的任意字符	\0	匹配NUL
\d	匹配数字	\r	匹配回车符
\D	匹配非数字字符	\n	匹配换行符
\s	匹配空白字符	\t	匹配制表符
\S	匹配非空白字符	\v	匹配垂直制表符
\w	匹配一个字母、数字或下划线等单词字符	\f	匹配换页符
\W	匹配非单词字符	[ ]	匹配方括号之间的任何字符,字符范围可以用"-"连接,例如[abcd]可以写为[a-d]
\b	查找处在单词的开始或结束处的匹配	[^]	匹配不在方括号之间的任何字符
\B	查找不在单词开始或结束处的匹配	(和)	指定子表达式,用于分组

在表5-2中,所谓"匹配",就是指子字符串符合某种条件(正则表达式),通常是指这个字符串中有一部分(或几部分分别)能满足正则表达式给出的条件。另外,空白字符是指空格符(space character)、回车符(carriage return character)、换行符(new line character)、制表符(tab character)、垂直制表符(vertical tab character)、换页符(form feed character)。

如果要查找元字符本身的话,比如查找字符"."或"*",应使用转义字符。转义字符使用"\"开始,用于取消紧跟在后面的字符的特殊意义。因此,要查找字符"."或"*",应该使用\.和*。要查找"\"字符本身,可使用\\。

例如:myfile\.dat匹配myfile.dat,D:\\GPMS3匹配D:\GPMS3。

在正则表达式中,还会遇到重复的情况,这需要使用量词(限定符)来指定重复的数量。常用的限定符见表5-3。

表 5-3　常用的限定符

量词/语法	说　　明
n *	匹配任何包含零个或多个 n 的字符串。例如：/lo * /g，在字符串中查找包含 0 个或多个“o”的字符串
n+	匹配包含至少一个 n 的任何字符串
n?	匹配任何包含零个或一个 n 的字符串
n{x}	匹配包含 x 个 n 的序列的字符串
n{x,y}	匹配包含 x 或 y 个 n 的序列的字符串
n{x}	匹配包含至少 x 个 n 的序列的字符串
^n	匹配任何开头为 n 的字符串。例如：/^You/g，查找所有以 Y 开头的字符串 You
n$	匹配任何结尾为 n 的字符串
? =n	匹配任何其后紧接指定字符串 n 的字符串。例如：/is(? =ok)/g，则查找所有的后面是“ok”的字符串“is”
?! n	匹配任何其后没有紧接指定字符串 n 的字符串

2. 创建 RegExp 对象

对于正则表达式，可以在有关的字符串对象中直接使用，也可以创建 RegExp 对象，然后在字符串操作中使用。前者使用直接量语法，一般形式为：

```
/pattern/attributes
```

如果要创建 RegExp 对象，一般形式为：

```
new RegExp(pattern, attributes);
```

其中，参数 pattern 是一个字符串，指定了正则表达式的模式或其他正则表达式。参数 attributes 是一个可选的字符串，包含属性“g”、“i” 和 “m”，分别用于指定全局匹配、区分大小写的匹配和多行匹配，gi 表示全局并区分大小写的匹配。

例如：str=str. replace(/\r/g,"
")；即为直接量语法，将字符串 str 中的所有回车用
替换。

下述代码创建了 RegExp 对象：

```
<script type = "text/javascript">
function checkemail(str)
{
    //在 JavaScript 中，正则表达式只能使用“/”开头和结束，不能使用双引号
var Expression = /\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;
var objExp = new RegExp(Expression);
if (objExp.test(str) == true)
  return true;
else
  return false;
}
</script>
```

3. RegExp 对象的属性和方法

RegExp 对象是 JavaScript 中提供的对字符串进行匹配、替换等操作的对象，和其他对象一样，也是由一系列的属性和方法构成，分别见表 5-4。

表 5-4 RegExp 对象属性

属 性	说 明
global	返回正则表达式是否具有标志“g”，查看给定的正则表达式是否执行全局匹配。如果 g 标志被设置，则该属性为 true，否则为 false
ignoreCase	返回是否设置“i”标志。如果设置了“i”标志，则返回 true，否则返回 false
multiline	返回正则表达式是否具有标志“m”。查看给定的正则表达式是否以多行模式执行模式匹配。在这种模式中，如果要检索的字符串中含有换行符，^和 $ 除了匹配字符串的开头和结尾外还匹配每行的开头和结尾
lastIndex	存放一个整数，保存上一次匹配文本之后的第一个字符的位置。上次匹配的结果是由方法 RegExp. exec()和 RegExp. test()找到的，它们都以 lastIndex 属性所指的位置作为下次检索的起始点。这样，就可以通过反复调用这两个方法来遍历一个字符串中的所有匹配文本。找不到可以匹配的文本时，它们会自动把 lastIndex 属性重置为 0
source	返回模式匹配所用的文本。该文本不包括正则表达式直接量使用的定界符，也不包括标志 g、i、m

RegExp 对象方法见表 5-5。

表 5-5 RegExp 对象方法

方 法	说 明
test(str)	方法 test()用于检测一个字符串是否匹配某个模式，参数 str 为要检测的字符串。如果字符串 str 中含有与 RegExpObject 匹配的文本，则返回 true，否则返回 false
exec(str)	exec() 方法用于检索字符串中的正则表达式的匹配，参数 str 为要检测的字符串。返回一个数组，其中存放匹配的结果。如果未找到匹配，则返回值为 null
compile(exp,mod)	compile() 方法用于在脚本执行过程中编译正则表达式。参数 exp 为正则表达式，mod 规定匹配的类型。“g”用于全局匹配，“i”用于区分大小写，“gi”用于全局区分大小写的匹配

例如，有下面的代码：

```
<script type = "text/javascript">
var str = "Hello World!";
var patt = /lo * /g;
document.write(str.match(patt));
</script>
```

则输出结果为：

```
l,lo,l
```

例如：(\d{1,3}\.){3}\d{1,3}是一个简单的 IP 地址匹配表达式。其中，\d{1,3}匹配

一到三位的数字,(\d{1,3}\.){3}匹配三位数字加上一个英文句号,这个整体被定义为一个分组,然后分组重复三次,最后再加上一个一到三位的数字(\d{1,3})。这是一个简单的IP地址模式匹配,当然,IP地址的每一个数字都是小于255的,可以进一步写出更加标准的验证IP地址的正则表达式。

在书写正则表达式时,还经常用[]来指定一个范围,例如:[0-9]代表的含意与\d完全一致,表示一位数字。[a-z0-9A-Z_]等同于\w。例如,对于下述代码:

```
<script type="text/javascript">
var str="Hello_你好!";
var patt=/\w/g;
document.write(str.match(patt));
</script>
```

输出结果为:

```
H,e,l,l,o,_
```

可见元字符"\w"只是匹配一个字母、数字或"_",不能匹配汉字等其他字符。

例如,正则表达式0\d\d-\d\d\d\d\d\d\d\d,匹配这样的字符串:以0开头,然后是两个数字,然后是一个连字号"-",最后是8个数字(即中国的区号为3位的电话号码)。其中"\d"是元字符,匹配一位数字(0,1,2,…)。"-"不是元字符,只匹配它本身——连字符或者减号。为了避免重复,也可以这样写这个正则表达式:0\d{2}-\d{8}。"\d"后面的{2}、{8}表示前面的"\d"必须连续重复匹配的次数为2次(8次)。

4. 常用正则表达式

正则表达式通常用于表单中的字符串处理、表单数据的有效性验证等,实用高效。下面是一些常用的正则表达式。

(1) 匹配中文字符的正则表达式:[\u4e00-\u9fa5]。

(2) 匹配双字节字符(包括汉字在内):[^\x00-\xff],可以用来计算字符串的长度(一个双字节字符长度计2,ASCII字符计1)。

(3) 匹配空白行的正则表达式:/\n\s*\r/,可以用来删除空白行。

(4) 匹配HTML标记的正则表达式:/<(\S*?)[^>]*>.*?</\1>|<.*?/>/。

(5) 匹配首尾空白字符的正则表达式:/^\s*|\s*$/,可以用来删除行首行尾的空白字符(包括空格、制表符、换页符等),非常有用的表达式。

(6) 匹配E-mail地址的正则表达式:/\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/,常用于表单验证。

(7) 匹配网址URL的正则表达式:/[a-zA-z]+://[^\s]*/。

(8) 匹配账号是否合法,以字母开头,允许5~16B,允许字母、数字、下划线,则符合上述要求的正则表达式为:/^[a-zA-Z][a-zA-Z0-9_]{4,15}$/。

(9) 匹配国内电话号码:/\d{3}-\d{8}|\d{4}-\d{7}|\d{4}-\d{8}/,匹配形式如010-11112222、0531-8836111或0531-88361111。

(10) 匹配中国邮政编码:/[1-9]\d{5}(?!\d)/,中国邮政编码为6位数字。

(11) 匹配身份证:/\d{15}|\d{18}/,中国的身份证为15位或18位。

(12) 匹配IP地址:/\d+\.\d+\.\d+\.\d+/,提取IP地址时有用。

(13) 匹配腾讯QQ号:/[1-9][0-9]{4,}/,腾讯QQ号从10000开始。

(14) 匹配特定数字,例如:

```
/^[1-9]\d*$/                  //匹配正整数
/^-[1-9]\d*$/                 //匹配负整数
/^-?[1-9]\d*$/                //匹配整数
/^[1-9]\d*|0$/                //匹配非负整数(正整数或0)
/^-[1-9]\d*|0$/               //匹配非正整数(负整数或0)
/^[1-9]\d*\.\d*|0\.\d*[1-9]\d*$/        //匹配正浮点数
/^-([1-9]\d*\.\d*|0\.\d*[1-9]\d*)$/   //匹配负浮点数
/^-?([1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.0+|0)$/  //匹配浮点数
/^[1-9]\d*\.\d*|0\.\d*[1-9]\d*|0?\.0+|0$/   //匹配非负浮点数(正浮点数 + 0)
/^(-([1-9]\d*\.\d*|0\.\d*[1-9]\d*))|0?\.0+|0$/  //匹配非正浮点数(负浮点数 + 0)
```

(15) 匹配特定字符串:

```
/^[A-Za-z]+$/        //匹配由26个英文字母组成的任意长度的字符串
/^[A-Z]+$/           //匹配由26个英文字母的大写组成的任意长度的字符串
/^[A-Za-z0-9]+$/     //匹配由数字和26个英文字母组成的字符串
/^\w+$/              //匹配由数字、26个英文字母或者下划线组成的任意长度字符串
```

5.5.3 Math 对象

JavaScript中的Math对象封装了常用的数学常数和一些常用的数学运算,这些运算包括三角函数、对数函数、指数函数和一些舍入函数等。

1. Math 对象的属性

Math对象中的属性与其他对象的属性有一些区别。这些属性是常用的数学常数,它们是定值,因此它们是只读的,不允许对这些对象属性进行写操作。表5-6列出了Math对象的属性名、描述和近似值。

表5-6 Math对象的属性

属性名	说明
E	常数e,称做欧拉常数,它是自然对数的底。近似值是2.718
LN2	2的自然对数,即以e为底的2的对数,近似值是0.693
LN10	10的自然对数,近似值是2.302
LOG2E	以2为底的常数e的对数,近似值是1.442
LOG10E	以10为底的常数e的对数,近似值是0.434
PI	π常数,即圆周长和直径之比,近似值是3.142
SQRT1-2	1/2的平方根,近似值是0.707
SQRT2	2的平方根,近似值是1.414

2. Math 对象的成员方法

Math 对象的成员函数可分为三角和反三角两类函数、对数、指数函数、舍入函数以及随机函数等，常用函数见表 5-7。

表 5-7 Math 对象的成员方法

成员函数	说明
sin(value)、cos(value)、tan(value)	求 value 的正弦、余弦和正切值，value 为弧度值
asin(value)、acos(value)、atan(value)	求 value 的反正弦、反余弦和反正切值，value 为弧度值
atan2(xvalue,yvalue)	直角坐标系中(xvalue,yvalue)点与 x 轴所成的角度
exp(value)	E 的 value 次方
log(value)	value 的以 e 为底的自然对数
pow(BaseValue,ExpValue)	BaseValue 的 ExpValue 次方
sqrt(value)	value 的平方根
abs(value)	value 的绝对值
ceil(value)	大于或者等于 value 的最小整数值
floor(value)	小于或者等于 value 的最大整数值
round(value)	value 四舍五入得到的整数值
random()	产生随机数

5.5.4 Date 对象

Date(日期)对象封装了有关时间和日期的一些变量和函数，利用这些变量，可以掌握当前日期和时间。Date 对象中没有一个属性是可以直接地设定或者取得的，这种思想更符合对象的封装思想，即成员变量都是私有的，提供公有的成员函数来访问私有的成员变量。

Date 对象的一般格式是：

```
Date(year, month, day, hours, minutes, seconds)
```

1. 创建日期对象

创建日期对象和数组对象有些相似，都需要使用 new 关键字，但它的构造函数比较复杂，有多个不同参数的构造函数，可以根据需要选择一种格式来生成一个新的日期对象，下面分别给出它们的语法和范例。

格式 1：`var DateObj = new Date();`

这是最简单的一种日期对象的创建格式。创建该对象时，激活默认的构造函数，该构造函数取计算机的当前时间作为日期对象的时间值。

格式 2：`var DateObj = new Date("月 日,年 小时: 分: 秒");`

在这种格式中，参数是一个字符串，描述了创建的 Date 对象的各个属性，该字符串需要满足以下情况。

(1) 在月、日之间的空格可以省略，但是在年、时间之间的空格不可以省略，否则会使日期无效。月份必须用英语的 January、February、…、December，不能用数字。

(2) 日、年之间的逗号不能省略。

(3) 时间部分的小时、分、秒间的冒号不可省略。

(4) 当“日”这一项超过当月的天数时，将自动进位换算到下一个月。“小时”这一项超过 24 时也将进位换算成日。例如：

```
var meetDate = new Date("June 22,2008 11:50:00");
```

格式 3：var DateObj = new Date(年,月,日);

这种格式中也有几点要说明：

(1) 参数是数值，不是字符串，年取后两位。

(2) 当“月”超过 12 或“日”超过当月天数时，将自动进位换算到下一年和月。

(3) “月”参数取值是从 0 到 11，即实际月份比参数大一。

(4) 小时、分、秒将被认为是 0。

格式 4：var DateObj = new Date(年,月,日,小时,分,秒);

这种格式与前一种很相近，例如：

```
var Date1 = new Date(96,2,23,20,55,00);
```

2. Date 对象常用方法

对于 Date 对象，不同的浏览器支持不完全相同。常用的 Date 对象方法见表 5-8。

表 5-8 Date 对象常用方法

方法名	功能
getDate()	从 Date 对象返回一个月中的某一天 (1 ～ 31)
getDay()	从 Date 对象返回一周中的某一天 (0 ～ 6)
getMonth()	从 Date 对象返回月份 (0 ～ 11)
getFullYear()	从 Date 对象以 4 位数字返回年份
getHours()	返回 Date 对象的小时 (0 ～ 23)
getMinutes()	返回 Date 对象的分钟 (0 ～ 59)
getSeconds()	返回 Date 对象的秒数 (0 ～ 59)
getTime()	返回 1970 年 1 月 1 日至今的毫秒数
setDate()	设置 Date 对象中月的某一天 (1 ～ 31)
setMonth()	设置 Date 对象中的月份 (0 ～ 11)
setFullYear()	设置 Date 对象中的年份(4 位数字)
setHours()	设置 Date 对象中的小时 (0 ～ 23)
setMinutes()	设置 Date 对象中的分钟 (0 ～ 59)
setSeconds()	设置 Date 对象中的秒钟 (0 ～ 59)
setTime()	以毫秒设置 Date 对象
toString()	把 Date 对象转换为字符串
toTimeString()	把 Date 对象的时间部分转换为字符串
toDateString()	把 Date 对象的日期部分转换为字符串

除了上述这些常用的方法外，Date 对象还包含许多关于世界时和本地时间的设置方法，在此省略。

【例 5-4】 编写程序，在页面上显示一个走动的时钟，并且当用户关闭页面时，显示页面在该页面停留的时间。

分析： 要获取页面的停留时间，对打开页面和关闭页面时间编程即可。

代码清单：

```
<html>
<head>
<script type="text/javascript">
var timeStr, dateStr;
var begintime = new Date();
function timeclock ()
{
  now = new Date();
  //时间
  hours = now.getHours();
  minutes = now.getMinutes();
  seconds = now.getSeconds();
  timeStr = "" + hours;
  timeStr += ((minutes < 10) ? ":0" : ":") + minutes;
  timeStr += ((seconds < 10) ? ":0" : ":") + seconds;
  document.myclock.mytime.value = timeStr;
  //日期
  day = now.getDate();
  month = now.getMonth() + 1;
  month = ((month < 10) ? "0" : "") + month;
  year = now.getYear();
  dateStr = "" + month;
  dateStr += ((day < 10) ? "/0" : "/") + day;
  dateStr += "/" + year;
  document.myclock.mydate.value = dateStr;
  Timer = window.setTimeout("timeclock()",1000);
}
function timeend()
{
  endtime = new Date();
  minutes = (endtime.getMinutes() - begintime.getMinutes());
  seconds = (endtime.getSeconds() - begintime.getSeconds());
  time = (seconds + (minutes * 60));
  alert("您在本页共停留了" + time + " 秒钟");
}
</script>
</head>
<body onload = "timeclock ()" onUnload = "timeend();">
  <form name = "myclock">
    时间: <input type = "text" name = "mytime" value = "" size = "10"><br>
    日期: <input type = "text" name = "mydate" value = "" size = "10">
  </form>
</body>
</html>
```

显示结果如图 5-7 所示。

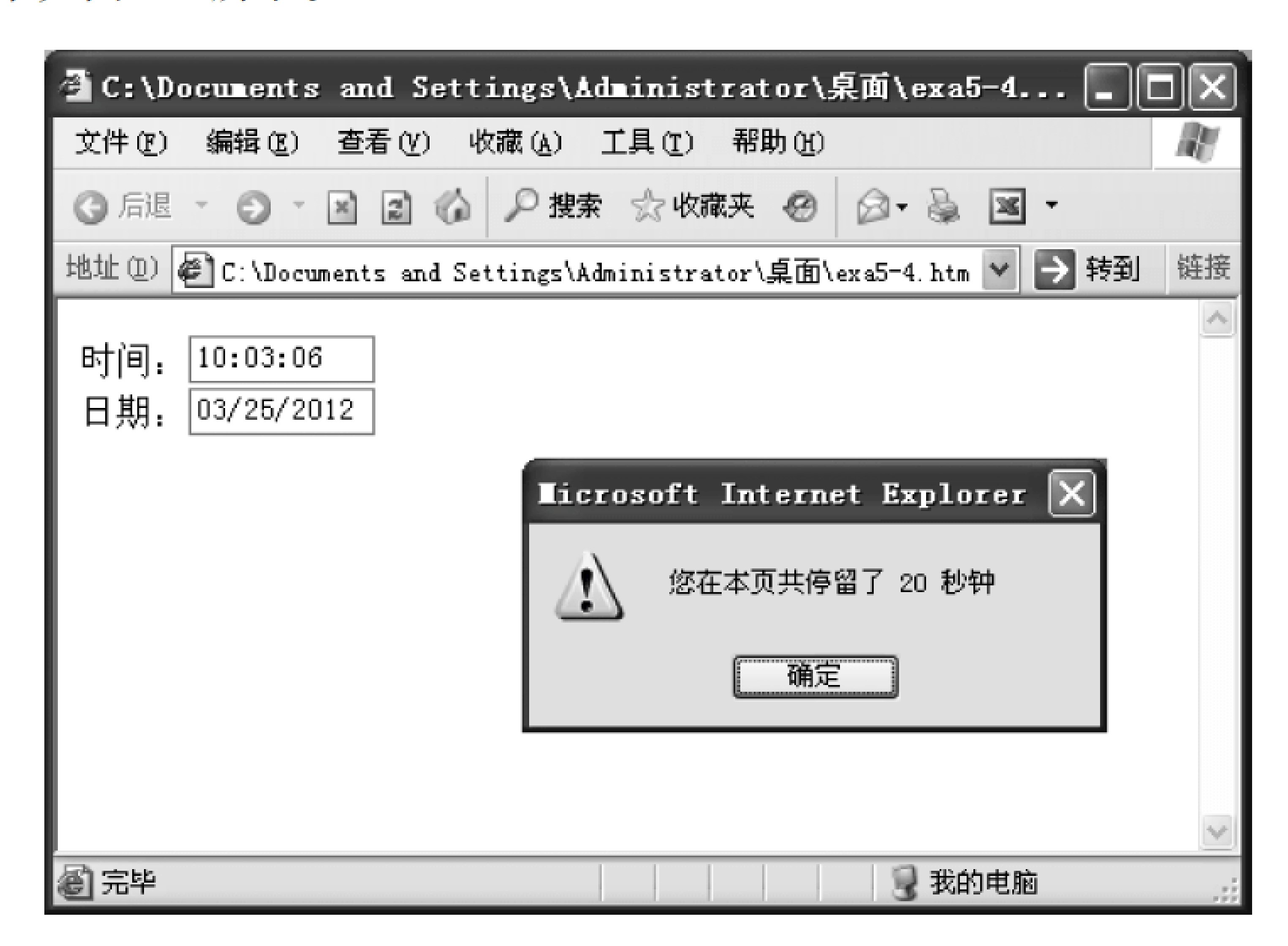


图 5-7 时钟显示示例

在上述程序中，用到了 window 对象、document 对象和 HTML DOM 对象，详细介绍见后面的几节。

5.5.5 Array 对象

与其他的高级语言不同，JavaScript 没有提供明显的数组类型。数组是一种内置的 Array 对象，它是一组元素对象的集合，元素可以是不同类型的。数组的每一个成员对象都有一个“下标”，用来表示它在数组中的位置(下标从 0 开始)。

1. 一维数组

在 JavaScript 中，要使用数组，必须创建 Array 对象。创建 Array 对象，有三种形式。

(1) 定义了一个空数组：

```
var <数组名> = new Array();
```

对于空数组，数组中元素的个数不确定。接下来可以为数组添加元素。

一般形式是：

```
<数组名>[<下标>] = <表达式>;
```

例如：

```
var colorArray = new Array();        //定义一个空数组
colorArray[0] = "red";
colorArray[1] = "green";
colorArray[2] = "blue";
colorArray[3] = 255;
```

(2) 创建一个确定元素数量的空数组:

```
var <数组名> = new Array(size);
```

参数 size 是期望的数组元素个数。返回的数组,length 字段将被设为 size 的值。

(3) 创建数组并初始化数组元素。

用户还可以在定义数组的时候直接初始化元素数据。一般形式是:

```
var <数组名> = new Array(<元素 1>, <元素 2>, <元素 3>, …);
```

当使用这些参数来调用构造函数 Array() 时,新创建的数组的元素就会被初始化为这些值。它的 length 字段也会被设置为参数的个数。

例如,var myArray = new Array(1, 4.5, "Hi "); 定义了一个数组 myArray,里边的元素分别是:myArray[0] == 1; myArray[1] == 4.5; myArray[2] == "Hi "。

但是,如果元素列表中只有一个元素,而这个元素又是一个正整数的话,将定义一个包含正整数个空元素的数组。例如,var myArray = new array(10),定义一个长度为 10 的数组,而不是一个数组,包含一个正整数元素 10。

2. 多维数组

JavaScript 只有一维数组,不能和一般的程序设计语言一样,使用类似 var myArray= new Array(3,4)的方法来定义 3×4 的二维数组,或者用 myArray[1,1]来访问“二维数组”中的元素。实际上,所谓多维数组,就是数组的元素本身也是一个数组。因此,要使用多维数组,在 JavaScript,可用下面的形式来定义:

```
var myArray = new Array(new Array(), new Array(), new Array(), …);
```

例如。要定义一个 3×4 的二维数组,定义如下:

```
var myArray = new Array(new Array(4), new Array(4), new Array(4));
```

然后,可以用 myArray[i][j]的形式访问“二维数组”中的元素。

在 JavaScript 中,数组中的元素可以是不同类型的,因此可以定义 oneArray = new Array(new Array(3), new Array(4)),它不是一个 3×4 的二维数组,实际上 oneArray 有两个元素,第一个元素是一个长度为 3 的数组,第二个元素是一个长度为 4 的数组。

3. Array 对象的属性

Array 对象常用的属性是 length 属性,保存数组的长度,即数组中元素的个数,它等于数组中最后一个元素的下标加 1。因此,想添加一个元素,可写做:myArray[myArray.length] =…。对于二维数组,例如上面的 myArray,document.write(myArray.length)将返回 3,而不是返回 3×4=12。也就是说 myArray 有三个元素,都是长度为 4 的数组。

4. Array 对象的方法

对于 Array 对象,不同的浏览器支持不完全相同。常用的 Array 对象方法见表 5-9。

表 5-9　Array 对象常用方法

方法名	功能
slice(<s>[, <e>])	返回一个从第 s 个元素到第 e 个元素的子数组,如果不给出 e,则返回的子数组从第 s 个元素到数组的最后一个元素
shift()	删除并返回数组的第一个元素
pop()	删除并返回数组的最后一个元素
push()	向数组的末尾添加一个或更多元素,并返回新的长度
unshift(e1,e2,…)	向数组的开头添加一个或更多元素,并返回新的长度
a. concat(a1,a2,…)	将数组 a1,a2,…中的元素添加到数组 a 中
reverse()	颠倒数组中元素的顺序
sort()	对数组的元素进行排序
toString()	把数组转换为字符串,并返回结果
join(<分隔符>)	把数组中的各个元素串起来,用<分隔符>作为元素之间的分割符,然后返回这个字符串

例如:

```
<script type = "text/javascript">
var aa = new Array(2)
aa[0] = "George"
aa[1] = "John"
var bb = new Array(3)
bb[0] = "James"
bb[1] = "Adrew"
bb[2] = "Martin"
document.write(aa.concat(bb))
</script>
```

在 JavaScript 中,可以直接输出一个数组对象,例如 document. write(aa),则输出数组中的每一个元素的值,元素值之间用逗号分开。

5.5.6　全局函数

除了内置对象外,还有一些功能是经常需要使用的,这些功能被定义为标准的内置函数,不属于任何对象,因此不用再通过引用对象的方式来使用它们,称为 JavaScript 全局函数。常用的全局函数见表 5-10。

表 5-10　JavaScript 全局函数

函数名	说明
parseInt(str, radix)	解析一个字符串,并返回一个整数。参数为 str 要被解析的字符串,参数 radix 表示要解析的数字的基数,介于 2～36 之间
parseFloat(str)	可解析一个字符串,并返回一个浮点数。只有字符串中的第一个数字会被返回。如果字符串的第一个字符不能被转换为数字,那么 parseFloat() 会返回 NaN
eval(string)	计算 JavaScript 字符串,并把它作为脚本代码来执行,如果有计算结果的话,则返回计算的结果

续表

函　数　名	说　　明
Number(object)	把对象的值转换为数字。如果参数是 Date 对象，Number() 返回从 1970 年 1 月 1 日至今的毫秒数。如果对象的值无法转换为数字，那么 Number() 函数返回 NaN
String(object)	把对象的值转换为字符串。参数 object 为 JavaScript 对象
isNaN(x)	检查是否非数字值。如果 x 是特殊的非数字值 NaN(或者能被转换为这样的值)，返回的值就是 true；如果 x 是其他值，则返回 false
isFinite(number)	用于检查其参数是否无穷大。如果 number 是有限数字(或可转换为有限数字)，那么返回 true；否则，如果 number 是 NaN(非数字)，或者是正、负无穷大的数，则返回 false
encodeURI(URIstr)	把字符串作为 URI 进行编码，返回 URIstring 的副本，其中的某些字符将被十六进制的转义序列替换。如果 URI 组件中含有分隔符，比如 ? 和 #，则应当使用 encodeURIComponent() 方法分别对各组件进行编码
decodeURI(URIstring)	对 encodeURI() 函数编码过的 URI 进行解码。返回值为 URIstring 的副本，其中的十六进制转义序列将被它们表示的字符替换
encodeURIComponent (URIstr)	把字符串作为 URI 组件进行编码。参数 URIstr 含有 URI 组件或其他要编码的文本
decodeURIComponent (URIstr)	对 encodeURIComponent() 函数编码的 URI 进行解码

【例 5-5】 JavaScript 字符串函数 eval()常常用于动态地得到变量名的操作，阅读下列代码，分析运行结果。

```
<script type="text/javascript">
var score1=97;
var score2=100;
for (i=0;i<2;i++)
{
   valstr="score"+(i+1);   //构造变量名
   document.write("valstr="+eval(valstr)+"<br>");
}
</script>
```

上述代码中，声明了两个整数变量，用循环输出它们的值，在循环中构造变量名，通过函数 eval(str)计算字符串 ctr 作为一个 JavaScript 表达式所对应的值。

5.6　浏览器对象

每一个符合 ECMAScript 规范的脚本程序设计语言都包含三个部分，即 ECMAScript 规范、浏览器对象模型(Browser Object Model，BOM)和文档对象模型(Document Object Model，DOM)。所谓浏览器对象模型(BOM)，就是指当用户打开浏览器时，浏览器中的 JavaScript Runtime Engine 将在内存中自动创建一组对象，用于对浏览器及 HTML 文档对象模型中数据的访问和操作。因为这些对象是和浏览器本身紧密相关的，称为浏览器对象。

5.6.1 浏览器对象模型

当使用浏览器打开一个网页时,浏览器就在内存中创建了一个 window 对象,它封装了浏览器的整个窗口,如果文档中包含框架(frame 或 iframe 标签),浏览器还会为每个框架创建一个额外的 window 对象。在 window 对象的封装中,包含窗口标题、工具按钮、地址栏、客户区、状态栏等,这些窗口的组成部分,也被定义为浏览器对象,它们都是 window 的成员对象,因此,构成一种层次结构,这就是浏览器对象模型。

JavaScript 定义的浏览器对象的层次结构如图 5-8 所示。

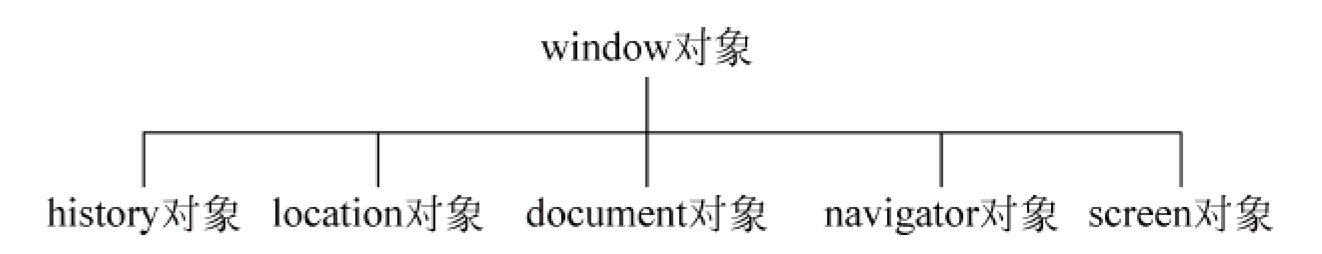


图 5-8 浏览器对象模型层次结构

在 JavaScript 中,浏览器对象模型(BOM)的 6 个对象中,window 对象是最顶层的对象,它对应浏览器窗口本身,其他 5 个对象均是 window 对象的成员对象,其中 document 成员对象也是 HTML 文档对象模型(DOM)中的重要对象。

浏览器对象模型中的每一个对象其实是一种预声明的内存对象,用户可以直接使用。需要注意的是,这些对象的名字的第一个字母都是小写的,写成大写就会出现 JavaScript 运行错误。

5.6.2 window 对象

window 对象表示一个浏览器窗口或一个框架。在客户端 JavaScript 中,window 对象是全局对象,所有的表达式都在当前的环境中计算。也就是说,要引用当前窗口不需要指定窗口对象本身,可以把 window 对象的属性作为全局变量来使用。例如,可以将 window.document 简写为 document,window.alert()可以简写为 alert()。这是对窗口对象的隐式应用,如果要显示地表明窗口,可写为 window.[属性|方法]或 self.[属性|方法]。对窗口中<frame>的引用见下面的 windows 属性介绍。

1. window 对象的常用属性

window 对象的属性可分为一般属性和对象属性,每一个属性具有不同的读写权限,即有的属性为只读属性,不能为属性进行赋值,有的属性则可以进行读写操作。window 对象的一般属性及成员对象见表 5-11。

表 5-11 window 对象属性表

属 性 名	说 明
frames	如果 window 中包含帧,则 frames 为一个数组对象,保存每个帧对应的子窗口。可以通过 window.frames["frame-name"]或数组下标 0~n 来访问

续表

属　性　名	说　　明
opener	返回对创建此窗口的窗口的引用
name	设置或返回存放窗口的名称的一个字符串。窗口的名称可以用做一个 <a> 或者 <form> 标记的 target 属性的值
window、self	window 和 self 均为对当前窗口的引用，相当于 this
top	返回顶级窗口的只读引用
closed	返回一个布尔值，该值声明了窗口是否已经关闭。该属性为只读。当浏览器窗口关闭时，表示该窗口的 window 对象不会消失，它将继续存在，不过它的 closed 属性将被设置为 true
location	location 对象是 window 对象的一部分，包含有关当前 URL 的信息。可通过 window.location 属性来访问
history	history 对象是 window 对象的一部分，包含用户在当前浏览器窗口中访问过的 URL
document	每个载入浏览器的 HTML 文档都会成为 document 对象，document 对象是 window 对象的一部分，document 对象使我们可以从脚本中对 HTML 页面中的所有元素进行访问
defaultStatus	设置或返回窗口状态栏中的默认文本，该属性可读可写
status	设置或返回窗口状态栏中的文本
navigator	navigator 对象包含有关浏览器的信息
screen	包含有关客户端显示屏幕的信息

【例 5-6】 编写一个网页文件 1.htm，确保不被嵌入到一个框架中打开。

分析：利用 window 对象的 self 和 top 属性可以判断窗口是否在一个框架中，如果是，则跳出框架。

代码清单：

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
</head>
<body>
<script type = "text/javascript">
if (window.top! = window.self){
  window.top.location = "1.htm"
}
</script>
</body>
</html>
```

【例 5-7】 有一个 Web 应用主界面采用帧来实现一个 1024 宽度的页面，并居中显示，在 1024 区域，又分成 200×10×＊三列，分别对应菜单、控制条和内容区。

代码清单：myframes.htm

```
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
</head>
<frameset cols = " * ,1024, * " framespacing = "0" frameborder = "0" id = "mainframes">
```

```
    <frame name="free-left" scrolling="no" noresize src="#">
    <frameset rows="75,20,*" framespacing="0" frameborder="0" id="frames01">
      <frame name="topframe" scrolling="no" noresize src="mybanner.html">
      <frame name="ctrltop"  scrolling="no" noresize src="scrolltop.html">
      <frameset cols="200,10,*" id="frames02">
        <frame name="leftframe" scrolling="auto" noresize src="menu.htm">
        <frame name="ctrlleft"  scrolling="no" noresize src="scrollleft.html">
        <frame name="mainFrame" scrolling="auto" noresize src="introduction.htm">
      </frameset>
    </frameset>
    <frame name="free-right" scrolling="no" noresize src="#">
</frameset>
</html>
```

要求完成控制条代码 scrollleft.html，使得单击控制条时，实现左侧帧的折叠和打开。

分析：这是一个多层嵌套的 window，需要注意的是，对于<frame>和<frameset>，不管它们处于哪个层次，都是顶层窗口 window.top 的成员对象，对象之间没有层次关系，这一点就如在<table>中使用<a>建立超链接，对<a>对象的访问不需要通过<table>对象一样。

代码清单：scrolltop.html

```
<html>
<head>
<script language="JavaScript">
var tt=1;
function ctrl_top()
{
   if (tt==1)
   {
     window.parent.frames01.rows="0,20,*";
     toparrow.src="toctrltop01.gif";
     tt=0;
   }
   else
   {
     window.parent.frames01.rows="75,20,*";
     toparrow.src="toctrltop02.gif";
     tt=1;
   }
}
</script>
</head>
<body style="margin:0px" onClick="ctrl_top()">
<table width="100%" border="0" cellspacing="0">
<tr>
  <td align="right">单击可显示或隐藏页首<img id="toparrow" src="toctrlleft02.gif"
align="absmiddle"></td>
</tr>
</table>
</body>
```

```
</html>
```

代码清单： scrollleft.html

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<script type="text/JavaScript">
var tt=1;
function ctrl_left()
{
   if (tt==1)
   {
     window.parent.frames02.cols="0,10,*";
     lrarrow.src="toctrlleft01.gif";
     tt=0;
   }
   else
   {
     window.parent.frames02.cols="200,10,*";
     lrarrow.src="toctrlleft02.gif";
     tt=1;
   }
}
</script>
</head>
<body style="margin:0px;background-color:#c0ddeb;cursor:hand" onClick="ctrl_left()">
<table height="100%" border="0" cellspacing="0">
<tr>
   <td align="center"><img id="lrarrow" src="toctrlleft02.gif"></td>
</tr>
</table>
</body>
</html>
```

上述代码清单的关键是访问<frameset>对象，在 JavaScript 运行时引擎为 HTML 标记创建 DOM 对象时，它们都是 window 对象的直接成员，直接通过 window 对象或标记的 id 属性即可访问。如果是要访问一个具体的<frame>，可以使用 window.frames["frame-name"]或 window.frame-name 来完成，其中 frame-name 是要访问的<frame>的 name 属性值。

例如：

```
window.parent.frames["frame-b"].document.bgColor = color;
```

可以在一个帧窗口 frame-a 中设置另一个帧窗口 frame-b 的背景颜色为 color 值。

2. window 对象的常用方法

window 对象的常用方法见表 5-12。

表 5-12 window 对象常用方法

方　　法	说　　明
open()	打开一个新的浏览器窗口或查找一个已命名的窗口，并返回该窗口对象的句柄。包含四个可选参数 URL，新窗口中要显示的文档的 URL，取值为空时，新窗口不显示任何文档 Name，新窗口的名称，可以用做标记 <a> 和 <form> 的属性 target 的值。如果该参数指定了一个已经存在的窗口，那么 open() 方法就不再创建一个新窗口，而返回对指定窗口的引用 Features，设置窗口特征 Replace，当取值为 true 时，URL 替换浏览历史中的当前条目；当取值为 false 时，URL 在浏览历史中创建新的条目
close()	关闭浏览器窗口，只有通过 JavaScript 代码打开的窗口才能够由 JavaScript 代码关闭。这阻止了恶意的脚本终止用户的浏览器
focus()	把键盘焦点给予一个窗口
blur()	把键盘焦点从顶层浏览器窗口移走，整个窗口由 window 对象指定。哪个窗口最终获得键盘焦点并没有指定
moveTo(x,y)	把窗口的左上角移动到一个指定的坐标(x,y)处
moveBy(x,y)	相对窗口的当前坐标把它在 x、y 方向移动指定的像素
resizeTo(width,height)	将窗口的宽度和高度调整为 width 和 height 个像素
resizeBy(width,height)	将窗口的宽度和高度增大或减少 width 和 height 个像素
scrollTo(xpos,ypos)	把内容滚动到指定的坐标处。参数 xpos/ypos 表示要在窗口文档显示区左上角显示的文档的 x/y 坐标
scroll(xnum,ynum) scrollBy(xnum,ynum)	将文档分别向右、向下滚动 xnum 和 ynum 像素
setInterval()	按指定周期(毫秒)调用函数或计算表达式。有两个参数。 code，必选参数，要调用的函数或要执行的代码串 millisec，周期性执行或调用 code 函数的时间间隔，以毫秒计
setTimeout()	在指定的毫秒数后调用函数或计算表达式。参数同 setInterval()，setTimeout() 只执行 code 一次。如果要多次调用，使用 setInterval() 或者让 code 自身再次调用 setTimeout()
clearInterval(id)	取消由 setInterval() 设置的 timeout。参数 id 必须是由 setInterval() 返回的 ID 值
clearTimeout(id)	可取消由 setTimeout() 方法设置的 timeout
alert(message)	显示带有一条指定消息和一个 OK 按钮的警告框
confirm(message)	显示一个带有指定消息和 OK 及“取消”按钮的对话框。如果用户单击“确定”按钮，则 confirm() 返回 true。如果单击“取消”按钮，则 confirm() 返回 false
print()	打印当前窗口的内容。类似用户单击浏览器中的“打印”按钮

对于 setInterval()和 setTimeout()两种方法，前者定义周期性执行的函数更加方便，不需要递归调用接口实现。例如，下列脚本程序可以在页面中显示一个走动的时钟：

```
<script type="text/javascript">
var clockid = self.setInterval("clock()",1000);
```

```
function clock()
{
  var nowtime = new Date();
  form1.clock.value = nowtime.toTimeString().substring(0,8);
}
</script>
```

当打开一个窗口时，可设置的窗口特征见表 5-13。

表 5-13 窗口特征

特 征	说 明
top、left	窗口左上角坐标的像素值
height、width	窗口的高度和宽度
titlebar	是否显示标题栏，取值为 yes|no|1|0。默认是 yes
menubar	是否显示菜单栏，取值为 yes|no|1|0。默认是 yes
toolbar	是否显示浏览器的工具栏，取值为 yes|no|1|0。默认是 yes
location	是否显示地址字段，取值为 yes|no|1|0。默认是 yes
status	是否添加状态栏，取值为 yes|no|1|0。默认是 yes
scrollbars	是否显示滚动条，取值为 yes|no|1|0。默认是 yes
resizable	窗口是否可调节尺寸，取值为 yes|no|1|0。默认是 yes
channelmode	是否使用剧院模式显示窗口，取值为 yes|no|1|0。默认是 no
fullscreen	是否使用全屏模式显示浏览器。默认是 no。处于全屏模式的窗口必须同时处于剧院模式
directories	是否使用剧院模式显示窗口，取值为 yes|no|1|0。默认是 no

【例 5-8】 当在浏览器中打开一个页面时，如果文档长度大于浏览器窗口的高度，则出现垂直滚动条。要求：编写代码，使得窗口滚动条置于窗口的底部。

分析： 这是在实际项目研发中常用的功能，它可能是单击上部一个按钮，在文档尾部添加了一条记录，应该自动地定位到文档底部。

代码清单：

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
<style>
body{
  margin:0px;padding:0px;
  overflow-x:hidden;
  overflow-y:hidden;
}
#myfixeddiv{
  position:absolute;
  top: 10;
  left:10;
  width:100%;
  background-color:#ff0000;
  border:1px solid #00ff00;
```

```
    z - index: - 1;
  }
  div # scrollcontent{
    width:100 % ;
    height:100 % ;
    overflow:auto;
    padding:0px 30px 30px 30px;
  }
  </style>
  < script type = text/javascript >
  function gobottom()
  {
    alert("scroll to bottom");
    //滚动窗口滚动条到页面底部,本处未用到
    var c = window.document.body.scrollHeight;
    window.scrollBy(0,c);
    //将 div 的滚动条移动到底部
    var divobj = document.getElementById('scrollcontent');
    divobj.scrollTop = divobj.scrollHeight;
  }
  </script>
  </head>
  < body style = "overflow - x:hidden;">
  < div id = "myfixeddiv">
  < input type = "button" value = "滚动条置底" name = "b1" onclick = "gobottom();">
  </div>

  < div id = "scrollcontent">
  < script type = "text/javascript">
  for (i = 1;i < 50;i++)
      window.document.write("Line" + i + "< br >");
  </script>
  < a href = " # " onclick = "window.scrollTo(0,0);document.getElementById('scrollcontent').
  scrollTop = 0;">回顶部</a>
  </div>
  </body>
  </html>
```

在浏览器中打开上述页面,显示结果如图 5-9 所示。

在上述代码中,有三个在实际项目研发中遇到的重要的技术难点,解释如下。

(1) 滚动条置底。通常情况下,滚动条会出现在窗口和 div 中,在窗口中,可以比较容易地控制滚动条的位置,即通过 gobottom()函数中的 window.scrollBy(0,c)即可实现。将图层中的滚动条定位到图层的底部,方法不同,见 gobottom()函数。

(2) 将图层固定,不随滚动条的滚动而滚动,且不闪烁。在正常的页面中,图层会随着滚动条的滚动而滚动,有些广告页面中,会看到广告窗口图层虽然定位在一个位置,但滚动条滚动时,图层会闪烁,影响效果。代码是将<body>分成了两个图层,一个输出固定不动的内容,另一个输出其他内容。

同时，设置<body>样式，将<body>的滚动条隐藏；设置<div>样式，显示滚动条，且图层宽度为100%，高度为100%，则输出内容使得<div>出现滚动条时，看上去类似窗口滚动条，移动，不影响另外的<div>，即使得图层固定不动，且不会闪烁。

（3）将固定图层的z-index设置为-1，使得在有多个图层叠加时，它在底部。

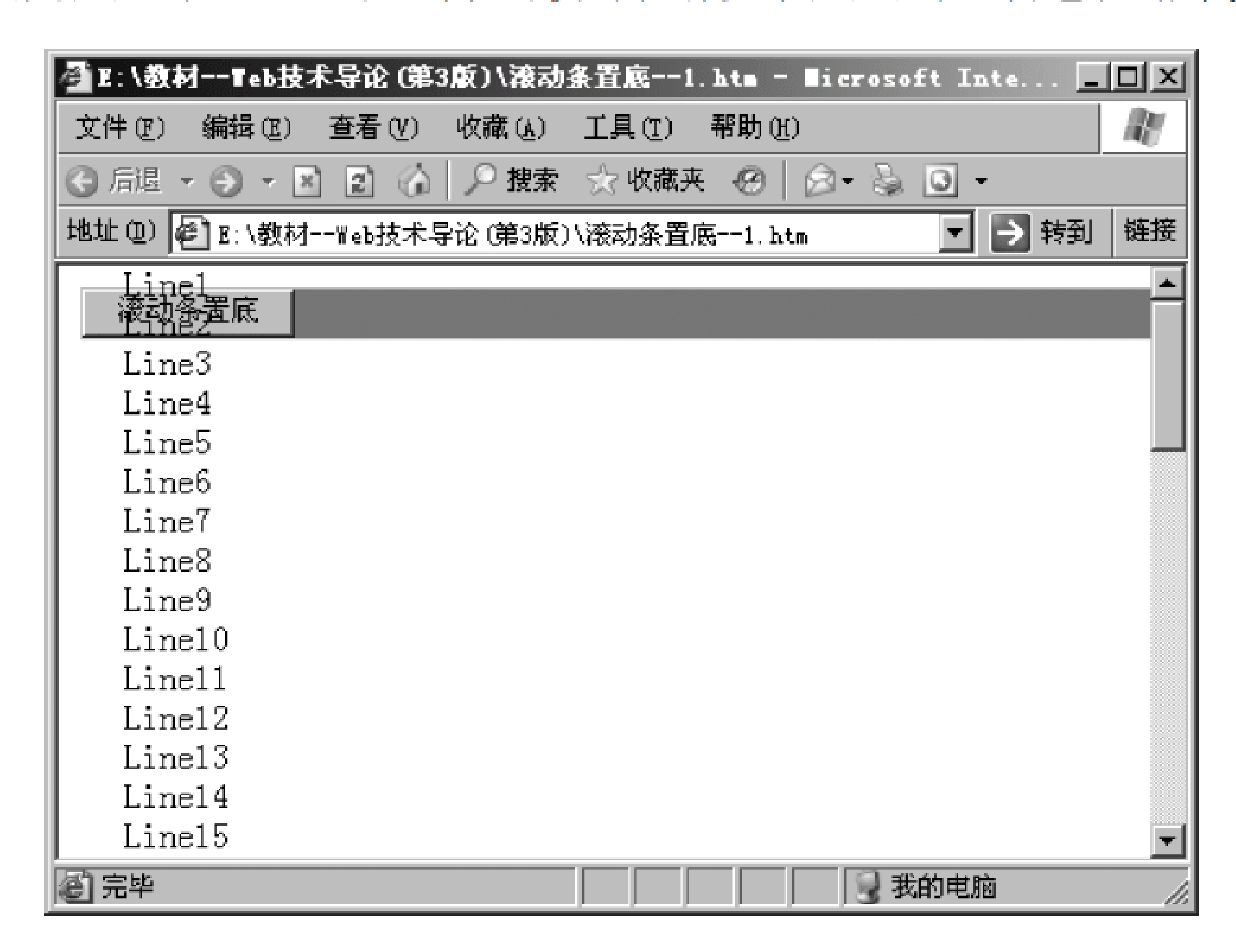


图5-9 滚动条置底操作页面

5.6.3 location对象

location对象是window对象的成员对象，封装了浏览器窗口的地址栏。location对象包含的属性见表5-14。

表5-14 location对象常用属性

属性名	说明
href	可读写属性，设置或返回完整的URL。可以通过为该属性设置新的URL，使浏览器读取并显示新的URL的内容
protocol	可读写属性，可设置或返回当前URL的协议，包括后面的冒号（:）
hostname	可读写属性，存储URL地址中的主机名+域名部分，不包括端口号
host	可读写属性，存储URL的主机名和端口号，只有端口号是URL的一个明确部分时，值中才包括端口号
port	可读写属性，设置或返回当前URL的端口部分
pathname	可读写属性，存储URL中文件路径。如果URL中的网页文件是根下的一个文件，则location.pathname的值为根（/）
hash	可读可写属性，该字符串是URL的锚部分（从#号开始的部分）
search	可读可写属性，设置或返回当前URL中的问号（“?”）之后的部分，即页面的参数部分

location 对象的方法见表 5-15。

表 5-15　location 对象常用方法

方法	说　　明
assign(URL)	把一个新的 URL 赋给当前窗口的 location 对象,即在当前窗口打开一个新的页面。也可以通过为 location. href 赋值来导航到一个新的网页。采用 assign 的方法会使代码易维护
reload([true|false])	如果没有参数,或者参数是 false,用 HTTP 头 If-Modified-Since 检测服务器上的文档是否已改变。如果文档已改变,reload() 会再次下载该文档。如果文档未改变,则该方法将从缓存中装载文档。这与用户单击浏览器中的"刷新"按钮的效果是完全一样的 如果该方法的参数设为 true,无论文档的最后修改日期是什么,它都会绕过缓存,从服务器上重新下载该文档。这与用户在单击浏览器中的"刷新"按钮时按住 Shift 键的效果是完全一样
replace(newURL)	用一个新文档取代当前文档。不会在 history 对象中生成一个新的记录。当使用该方法时,新的 URL 将覆盖 history 对象中的当前记录

我们在介绍 document 对象时曾提及一个 location 属性,我们要将这两个同名者区分开。只需要记住 location 对象是可读可写的,就是说,可以对 location 对象的属性赋值;而 document 对象的 location 属性是一个只读属性,它只是提供文档的 URL,改变它也是没有意义的。因为,document 的 location 属性描述了文档的 URL,改变它就是表示这个文档被复制到另外的 URL,这是不可能发生的。

例如:下面的代码定义一个超链接,单击,则在当前窗口打开一个新的网页。

```
<a href = "#" onclick = "window.location.assign('http://www.google.com')"> Google 搜索</a>
```

【例 5-9】 利用 location 对象,在 htm 页面中,编写一个函数获取并输出传入的参数名及参数值。

分析: 在 HTML 中,当打开一个网页时,经常会传入相应的参数,如果调用的是服务器页面(如 JSP 页面),则服务器页面可以通过服务器端的 request. getParameter("id");程序来获取页面中的参数,这和表单数据的获取相同。如果调用的是 htm 文档,则需要通过客户端的 location 对象来获取,并且需要使用类。

假设有两个页面 1. htm 和 2. htm,在 1. htm 中,定义了一个超链接:

```
<a href = "2. htm?username = jane&age = 18" target = "_blank"> test parameter </a>
```

要在客户端实现上述功能,需要在 2. htm 中定义一个求传入参数的类,来解析 URL 中的参数表,然后将解析后的参数名称和参数值,在当前页面创建为内存变量。

代码清单:

```
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
<script type = "text/javascript">
function GetParaString()
{
  var name, value, i = 0;
```

```
    var str = window.location.href;                    //取浏览器地址栏 URL 串
    var num = str.indexOf("?")
    str = str.substr(num + 1);                         //截取"?"后面的参数串
    var arrtmp = str.split("&");                       //将各参数分离形成参数数组
    for (i = 0;i < arrtmp.length;i++)
    {
       num = arrtmp[i].indexOf(" = ");
       if (num > 0) {
           name = arrtmp[i].substring(0,num);          //取得参数名称
           value = arrtmp[i].substr(num + 1);          //取得参数值
           this[name] = value;                         //定义对象属性并初始化
       }
    }
  }

  function showData(obj)
  {
    for(var prop in obj)
       document.write(prop + ":" + obj[prop] + "<br>");
  }
  </script>
  </head>
  <body>
  <p>传入参数为:</p>
  <script type = "text/javascript">
    var Request = new GetParaString();                 //创建参数对象实例
    var myname  =  Request["username"];                //取参数 username
    var myage   =  Request["age"];                     //取参数 age
    myage  =  parseInt(myage,10);                      //将年龄字符串转化为整数
    showData(Request);
  </script>
  </body>
  </html>
```

上述代码完成后,进行测试:在 1.htm 中,单击超链接 test parameter,则打开一个新窗口,显示 2.htm,在页面的顶部,显示传入的参数名和参数值。

5.6.4 history 对象

history 对象是 window 对象的成员对象,它保存了当前窗口或框架在某时间段内曾经打开网页的 URL 列表。使用 history 对象只能在这些列表之间跳转,不能得到具体的 URL 串。出于安全方面的原因,history 对象并不向脚本提供列表的 URL 串,也就是说无法通过 history 对象来得到用户曾经访问的 URL 的具体值。这是为了防止恶意的脚本程序获得用户个人的浏览习惯而进行不正当的网络传输或散播,这些信息是大部分用户不愿意泄露的。

对于 history 对象,只有一个属性,即 length,它是只读属性,返回当前窗口历史清单的长度。history 对象方法见表 5-16。

表 5-16 history 对象的方法

方 法	描 述
back()	装载历史清单中的前一个 URL,相当于工具栏中的“后退”按钮
forward()	装载历史清单中的后一个 URL,相当于工具栏中的“前进”按钮
go(x)	参数既可以是整数,也可以是字符串。当参数是整数 x 时,装载历史清单中当前位置偏移 x 后的 URL。当参数是字符串时,装载历史清单中含有这个字符串的最近 URL

如果我们的网页有很多内容,那么靠浏览器提供的“后退”按钮返回前一个 URL 较慢,可以在网页中设置“快速返回”按钮。代码如下:

```
<input type = "button" value = "快速返回" onClick = "history.go(-5)">
```

在每页中的适当位置设置这个按钮,每单击一次将向后 5 个 URL,结合浏览器中的“后退”按钮,将可以快速地向后翻动。

5.6.5 screen 对象

screen 对象是 window 对象的成员对象,它封装了用户计算机显示屏幕的信息。JavaScript 程序可以利用这些信息来优化输出,以达到用户的显示要求。screen 对象常用属性见表 5-17。

表 5-17 screen 对象属性

属 性	描 述
availHeight availWidth	存储浏览器屏幕可用的高度和宽度的像素数
height width	存储浏览器屏幕的高度和宽度的像素数

除了上述属性外,screen 对象还有一组与分辨率、色彩有关的属性,在此省略。此外,和其他对象不同,screen 对象没有提供方法。

由于现在的大多数浏览器采用了标签式页面管理,不再是 IE 6.0 那样,打开一个网页,就打开一个浏览器窗口,它们都体现在 Windows 操作系统的任务栏中。现在,这些页面窗口都被组织在浏览器窗口中,这使得页面的切换和关闭变得容易。

在标签式页面组织模式下,如果在一个窗口通过 window.open()方法动态地打开一个窗口,不管如何设置新建窗口的属性,或者移动新窗口的位置,以及对窗口进行放大、缩小等操作,将都不生效,而是和其他窗口一样作为一个页面标签,出现在浏览器窗口中。

5.6.6 navigator 对象

在 BOM 中,navigator 对象是 window 对象的成员对象,它封装了有关操作系统、浏览器版本等环境信息。navigator 对象最早是 Netscape 提出的,但其他实现了 JavaScript 的浏览器同样支持这个对象。

navigator 对象常用属性及方法见表 5-18。

表 5-18　navigator 对象属性及方法

属　性	
plugins[]	plug-in 对象的数组，其中的元素代表浏览器已经安装的插件。plug-in 对象提供的是有关插件的信息，其中包括它所支持的 MIME 类型的列表
appName	存储表示浏览器名称的字符串
appVersion	存储客户所用浏览器的版本号和操作系统等信息，不同的浏览器显示的内容项目不同
appMinorVersion	浏览器的次级版本
appCodeName	浏览器代码名称
platform	返回运行浏览器的操作系统平台
方　法	
javaEnabled()	该方法可返回一个布尔值，该值指示浏览器是否支持并启用了 Java。如果是，则返回 true，否则返回 false

5.7　HTML 文档对象

当浏览器打开一个网页时，不管是 HTML 还是 XML 文档，对于网页中的每一个标记，都在内存中创建一个相应的内存对象。这些对象按照树形结构组织，这就是文档对象模型(DOM)。可以将 DOM 理解为网页的 API，它将网页中的元素看做一个个对象，这些对象通过标记的 name 属性来命名，通过对这些可访问的内存对象进行编程，来实现对网页中元素及其属性的修改，以便动态地修改网页。

5.7.1　文档对象模型

根据 W3C DOM 规范，文档对象模型(DOM)是一种与系统平台、浏览器、语言无关的接口。DOM 解决了 Netscape 的 JavaScript 和 Microsoft 的 JScript 之间的冲突，为 Web 应用开发提供了一个标准的方法，从而使应用程序和浏览器脚本能动态地访问和更新文档的内容。

W3C DOM 被分为三个不同的部分，即核心 DOM、XML DOM 和 HTML DOM。核心 DOM 是用于任何结构化文档的标准模型，XML DOM 和 HTML DOM 分别是用于 XML 文档和 HTML 文档的标准模型。在 DOM 中，定义了所有文档元素的对象和属性，以及访问它们的方法。

对于 DOM 对象的访问，可以出现在页面的脚本程序中，也可以直接在浏览器地址栏中书写。例如，当浏览网页时，有时候需要知道网页的发布时间，从而判断网页内容是否很久以前的，此时，可以在浏览器的地址栏中输入：

```
javascript:document.write(document.lastModified)
```

输入结束后，按 Enter 键，则打开一个新的网页显示当前正在浏览的网页的最后修改日期。然后单击浏览器工具栏中的“后退”按钮，返回到刚才浏览的网页。

1. HTML DOM 对象

在 HTML 文档中,当浏览器打开一个网页时,JavaScript 运行时引擎将为每一个标记在内存中建立一个内存对象,即 HTML DOM 对象。根据 HTML 的层次结构,这些 HTML DOM 对象也表现为一种层次关系,形成 HTML DOM 树。

根据 HTML DOM 的概念,每一个 DOM 对象都对应一个 HTML 标记,因此,常用的 HTML DOM 对象见表 5-19。

表 5-19 常用 HTML DOM 对象

DOM 对象	说明
document	表示整个 HTML 文档,可用来访问页面中的所有元素
meta	对应一个<meta>元素
Link	对应一个<link>元素
style	对应一个单独的样式声明
base	对应<base>元素
body	对应<body>元素
Image	对应<img>元素
Anchor	对应<a>元素
Table	对应<table>元素
TableRow	对应<tr>元素
TableData	对应<td>元素
area	对应图像地图中的<area> 元素
frameset	对应<frameset>元素
frame	对应<frame>元素
iframe	对应<iframe>元素
form	对应<form>元素
button	对应<button>元素
input radio	对应表单中的一个单选按钮
input text	对应表单中的一个文本框
input password	对应表单中的一个密码域
textarea	对应<textarea>元素
input checkbox	对应表单中的一个复选框
select	对应表单中的一个选择列表
option	对应<option>元素
input file	对应表单中的一个文件上传
input hidden	对应表单中的一个隐藏域
input submit	对应表单中的一个"确认"按钮
input reset	对应表单中的一个"重置"按钮
object	对应<object>元素
Event	代表某个事件的状态

在表 5-19 中,在 DOM 对象列,有的是可以使用的对象,例如 document、body 对象,因为一个 HTML 文档只有一个 document 和一个 body 对象;有的是对象类,例如 Image,因为一个 HTML 文档可能包含多个<img>标记,每个标记都创建一个 Image 对象。所有的

HTML DOM 对象之间为层次结构关系，这种关系和 HTML 规范中标记之间的层次关系一致。因此，可以说 document 对象是所有其他 DOM 对象的父对象，其他对象都是通过 document 对象访问的。HTML DOM 对象层次结构关系如图 5-10 所示。

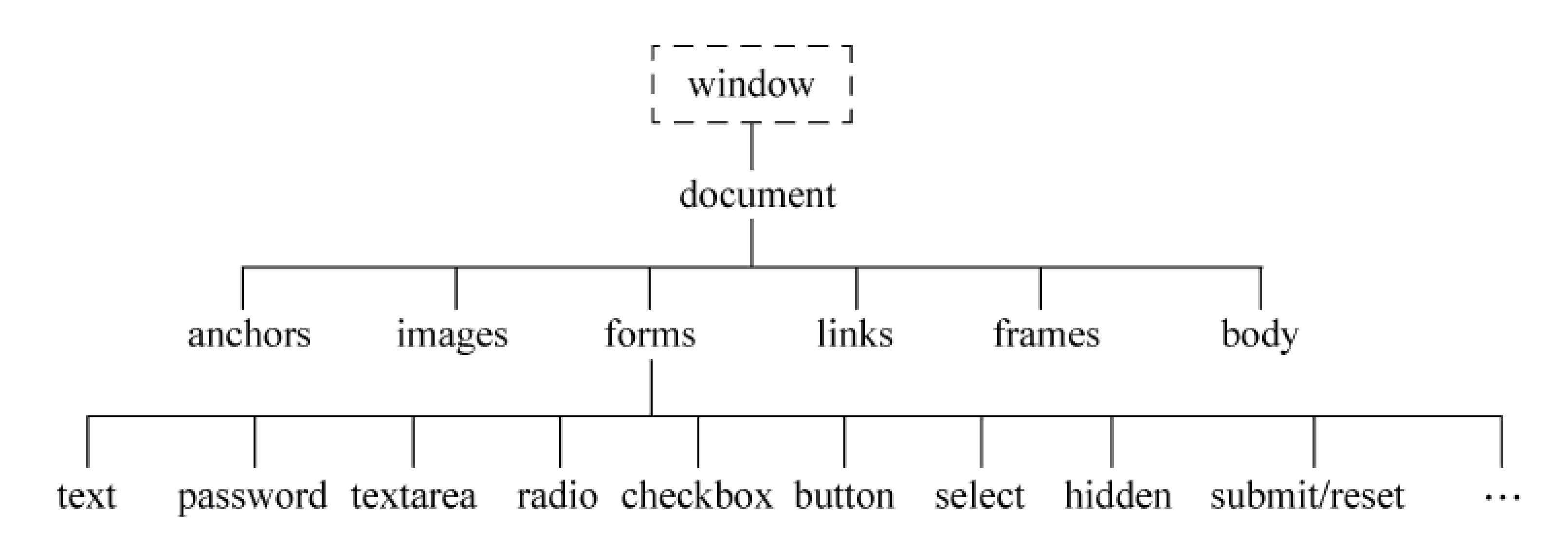


图 5-10　HTML DOM 对象层次结构

2. DOM 对象属性及方法

记住每一个 HTML DOM 对象的属性和方法是非常困难的。但是只要理解了 HTML 标记和 HTML DOM 对象之间的关系，则 DOM 对象的学习将变得轻松。因为，在 HTML DOM 中，每一个对象对应一个 HTML 标记，HTML 标记的属性则对应着 HTML DOM 对象的属性和方法。HTML 标记的一般属性对应 HTML DOM 对象的属性，HTML 标记的事件属性对应 DOM 对象的方法。可见，当我们熟悉了 HTML 标记的属性后，HTML DOM 对象就很容易理解了。

例如：对应<img src= "smile.jpg" style="position:absolute; bottom:0px">，将创建一个 image 对象，该对象必然有一个 src 属性和 style 属性，且 src 属性的值为 smile.jpg，style 属性将图片定位在窗口的底部。通过修改 image 对象的属性值，从而对<img>标记的图片进行操作。

5.7.2　document 对象

在 HTML DOM 中，document 对象代表整个 HTML 文档，可用来访问页面中的所有元素，包括文档头<head>部分和文档体<body>部分。此外，需要注意的是 document 对象是浏览器对象模型(BOM)中的 window 对象的一个成员对象，可通过 window.document 属性来访问。

1. document 对象属性

在 HTML DOM 中，document 对象封装了整个 HTML 文档，因此文档的每一个部分都可以通过 document 对象来访问，文档中所有内容也被封装成了 document 对象的属性。HTML 文档是由文本、图片、超链接、表格、表单、图层、span、嵌入对象、脚本程序等构成的，它们构成了 document 对象的属性。document 对象属性名见表 5-20。

document 对象包含的属性很多，在 FrontPage 中的代码视图，输入 window.document，可打开 IntelliSense 窗口，显示其包含的全部属性和方法。

表 5-20　document 对象常用属性

属　性　名	说　　明
all[]	提供对文档中所有 HTML 元素的访问
images[]	提供对文档中所有 Image 对象的引用
anchors[]	返回对文档中所有 Anchor 对象的引用，每一个 anchor，对应一个超链接标记<a name="bookname">，如果含有 href 属性，则对象同时也保存到 links[]数组中
links[]	返回对文档中所有 Link 对象和<a href="url">超链接的引用
forms[]	返回对文档中所有 Form 对象的引用
body	提供对 <body> 元素的直接访问。对于定义了框架集的文档，该属性引用最外层的 <frameset>
title	返回当前文档的标题
lastModified	返回文档被最后修改的日期和时间
cookie	设置或返回与当前文档有关的所有 cookie
domain	返回当前文档的域名
URL	返回当前文档的 URL
referrer	返回载入当前文档的 URL

2. document 对象方法

document 对象的方法见表 5-21。

表 5-21　document 对象常用方法

方　　法	描　　述
open(mimetype,replace)	打开一个输出流，返回一个新的 document 对象，该对象可以接收来自 write()或 writeln()方法的输出 参数 mimetype，可选，规定正在写的文档的类型，默认值是 text/html 参数 replace，新文档将覆盖当前的网页内容
write(exp1,exp2,…)	向文档写 HTML 表达式或 JavaScript 代码，如果是一个逻辑表达式，则输出 true 或 false，如果是一个 Array 对象，在输出数组元素的值，多个元素值之间用逗号分开
writeln(exp1,exp2,…)	等同于 write()方法，输出结束后输出一个换行符
close()	关闭用 document. open()方法打开的输出流，并显示选定的数据
getElementById(id)	返回对拥有指定 id 的第一个对象的引用
getElementsByName (name)	返回带有指定名称的对象集合。另外，因为一个文档中的 name 属性可能不唯一(如 HTML 表单中的单选按钮通常具有相同的 name 属性)，所以 getElementsByName() 方法返回的是元素的数组，而不是一个元素
getElementsByTagName (tagname)	返回带有指定标签名的对象集合，返回元素的顺序是它们在文档中的顺序

HTML DOM 定义了多种查找元素的方法，除了 getElementById() 之外，还有 getElementsByName() 和 getElementsByTagName()。

不过，如果需要查找文档中的一个特定的元素，最有效的方法是 getElementById()。在操作文档的一个特定的元素时，最好给该元素一个 id 属性，为它指定一个(在文档中)唯一的名称，然后就可以用该 ID 查找想要的元素。可以用 getElementsByTagName() 方法

获取任何类型的 HTML 元素的列表。例如,下面的代码可获取文档中所有的表:

```
var tables = document.getElementsByTagName("table");
alert ("This document contains " + tables.length + " tables");
```

3. 访问 HTML DOM 对象方法

利用文档对象 document,可以用多种方法获得一个 HTML 标记对应的 DOM 对象,主要有以下几种方法。

1) 利用标记的 id 属性访问对象

在同一个 HTML 文档中,标记的 id 属性应该是唯一的,因此可以通过 id 来访问对应的内存 DOM 对象。有两种方法。

方法 1:获取 id 对应的 DOM 对象。

```
var obj = document.getElementById("element - id");
```

方法 2:直接使用 id 标识。

```
element - id.[属性|方法] = …
```

如果有一个标识元素 id 的字符串,使用 eval()函数,可以得到对应的 DOM 对象,即:

```
Obj = eval("element - id")
```

这种转化在编程时也很重要。

2) 利用标记的 name 属性访问对象

JavaScript 运行时引擎在创建 DOM 对象时,标记的 name 属性即成为 DOM 对象的名字。通过 name 属性访问对应的内存对象,有两种方法。

方法 1:获取 name 对应的 DOM 对象。

```
var obj = document.getElementsByName("element-name");
```

因为一个 HTML 中,元素的 names 属性可以重名,因此返回的可能是一个数组。

方法 2:直接使用元素的 name 值。

对于在<form>中设计的输入域,通常会给定一个 name 属性,此时可以使用下列形式访问表单中的这些输入域对象:

```
obj = document.form - name.element - name
```

也可以使用

```
obj = document.all.element - name 或 obj = document.all("element - name")
```

对于<form>外的元素标记,不能通过元素的 name 属性直接访问对象。例如:对于一个,则下列两种用法都是错误的。

```
document.sp1.innerText = "xxx",
document.all.sp1.innerText = "xxx"
```

要使用上述写法,需将<span>的 name 属性设置改为 id 属性设置,即<span id=

"sp1">。

3）利用 document 的 all 成员对象访问

所有的 DOM 对象，都保存在 document 的 all、images、links、anchors 和 forms 数组成员中，利用这几个成员可以访问 DOM 对象。数组元素的访问可以利用对象 id、对象名或元素下标 0…n 进行。

使用 all 成员对象访问 DOM 对象有多种形式：

```
obj = document.all.element - id
obj = document.all["element - id"]
obj = document.all("element - id")
obj = document.all.item("element - id")
```

需要注意的是，必须使用元素的 id 属性，不能使用 name 属性，因为在一个 HTML 文档中，name 属性是可以重名的，重名时，JavaScript 会创建一个数组，而 id 属性是不能重名的，它唯一地标识了一个元素。

【例 5-10】 正确访问 HTML DOM 对象是进行 JavaScript 编程的基础，阅读下列代码，分析运行结果。

代码清单：accessdoms.htm

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
<style type = "text/css">
span{font-size:13px; color: #0000FF;cursor:hand}
</style>
<script type = "text/javascript">
function imgmove(strid)
{
    obj = document.getElementById(strid)
    obj.style.bottom = 100;
    eval(strid).style.bottom = 100;
}

function winclose(obj)
{
   document.form1.btnclose.value = obj.innerHTML;
}

function divdisplay(strid)
{
   obj = document.all[strid];
   obj.style.left = (document.body.clientWidth-100)/2;
   obj.style.top = (document.body.clientHeight-75)/2;
   if (document.all.item(strid).style.display == "none")
      document.all.item(strid).style.display  =  "block";
   else
      document.all.item(strid).style.display  =  "none";
}
```

```
</script>
</head>
<body>
<span onclick = "imgmove('dogimg')">访问 image 对象</span>||<span onclick = "winclose(this)">
访问对象与参数传递</span>||<span onclick = "divdisplay('popupdiv')">显示或隐藏图层</span>
<img src = "mydog.jpg" id = "dogimg" style = "position:absolute;bottom:0;left:0">
<div id = "popupdiv" style = "position:absolute;top:50;left:50;width:100;height:75;border:
1px solid #ff0000;padding:10px;text-align:center;display:none">
图层在客户区居中
</div>
<form name = "form1">
<input type = "text" name = "btnclose" value = "">
</form>
</body>
</html>
```

对于上述代码清单，说明如下。

（1）对图片对象、图层和表单对象，分别使用了不同的方法进行访问和操作。但都是为了获取这个 DOM 对象。

（2）对于 JavaScript 函数定义和调用，形式参数可以是字符串，也可以是一个对象，两者在函数内部使用时不同。比如：imgmove(strid)函数的定义，在实际函数调用时，参数使用的是字符串，因此在函数内部，通过 eval()，将参数字符串转化成对应的对象。函数 winclose(obj)在调用时，参数为 this，即当前对象，因此函数内部直接使用参数名，而不能转换。

在定义 JavaScript 函数时，如果某个参数为标记的 id，在函数调用时，如果实际参数只写 id 号，则参数代表的是一个对象，如果参数写成 id 号两侧用双引号或单引号括起来，参数则是字符串，两者有着很大的不同，直接影响函数内部的语句写法。

（3）我们使用<span>模拟了一个超链接，这也是<span>非常灵活的地方，特别是可以对<span>任意地定义样式。

网页 accessdoms.htm 在浏览器中的显示结果如图 5-11 所示。



图 5-11　对象访问示例页面显示结果

对于 document 对象,在执行输出操作后,最后一定要使用 close()方法关闭输出流,否则,在浏览器的状态栏将显示蓝色进度条,显示下载未完成。例如,在一个数据记录内容修改页面,向一个 iframe(name= filelistbox)中输出内容,代码如下:

```
document.filelistbox.document.write("<body style = 'margin-top:0px;margin-left:10px;font-size:12px;line-height:150%'>");
document.filelistbox.document.write("<span style='display:none;'>" + iframestr + "</span>");
document.filelistbox.document.write("…");
…
document.filelistbox.document.write("</body>");
document.filelistbox.document.close();
```

5.7.3 body 对象

每一个 HTML 文档有一个<body>标记,浏览器在内存中创建一个 body 对象,body 对象封装了 HTML 文档中<body>标记的所有属性及所有的文档体元素。从 HTML 结构看,body 对象是 document 对象的成员对象。body 对象的常用属性见表 5-22。

表 5-22 document.body 对象属性

属 性	说 明
id	设置或返回 body 的 id
className	设置或返回<body>元素的 class 属性
dir	设置 <body> 元素的文本方向,取值为 rtl
topMarign、bottomMargin leftMarign、rightMargin	存储浏览器窗口客户区内容和上、下、左、右边框的距离(像素)
clientLeft、clientTop	存储浏览器窗口客户区左上角(x,y)坐标
clientWidth、clientHeight	存储浏览器窗口客户区宽度和高度
innerHTML	innerHTML 可以赋值一段符合 HTML 规范的文本,该文本将重写现有网页的<body>体内的内容
innerText	为 innerText 赋值,页面将用 innerText 的内容重写<body>体内的内容
bgColor	设置或返回<body>元素的背景色属性
disabled	如果该成员变量设为 true,则页面的内容被灰化,且不可被选取,右击时快捷菜单被禁用

body 对象的 innerHTML 和 innerText 的属性功能不同,如果一段字符串文本包含 HTML 标记,将该文本赋给 innerText,则页面内容用 innerText 的文本替换,该文本不按照其包含的 HTML 标记显示;如果同样的文本赋给 innerHTML 属性,则页面内容被 innerHTML 中的文本替换,该文本按照 HTML 规范解析。例如:document.body.innerText=" Hello "和 document.body.innerHTML=" Hello ",产生的结果不同。

从 HTML 文档结构看,body 对象是 document 对象的成员对象,对文档体内所有元素的访问应该通过 document.body 来进行。但是,为了书写上的方便,在 document 对象中,直接包含了有关文档体内的元素对象。例如:设置文档的背景色,理论上应该写为

document. body. bgColor="",但是,也可以直接写为 document. bgColor="",两种写法结果是一样的。实际上,对文档体中大多数元素的访问几乎都可以直接使用 document,而不是 document. body。

5.7.4 image 对象

在 HTML 文档中,每个<img>标记都创建一个 image 对象,这些对象保存在 document 对象的 images 数组成员中。因为<img>标记的属性很多,这也就决定了 image 对象的属性。image 对象的属性见表 5-23。

表 5-23 image 对象属性

属　　性	说　　明
id	设置或返回图像的 id
name	设置或返回图像的名称
src	image 对象对应图片的 URL
height、width	图片的高度和宽度
vspace	图片在垂直方向上与上面或下面文字之间的距离
border	图片周围边框的宽度,默认值为 0
align	可设置返回与内联内容的对齐方式
className	设置或返回元素的 class 属性
title	设置或返回元素的提示性标题

在网页中使用图片,除了增加页面的视觉效果,还可以通过 JavaScript 程序动态地选择载入的图片,或者实现一些动画效果,从而增加页面的动感。

【例 5-11】 有两幅小狗左右张望的图片,利用 image 对象,实现简单的动画效果。

分析: 只要利用 window 对象的 setInterval()函数,两幅图像轮流显示即可。

代码清单: imgloop. htm

```
<html>
<head>
<script type = "text/javascript">
var num = 1;
var imgslist = new Array(2);
for (i = 0;i < 2;i++)
{
  imgslist[i] = new Image();
  imgslist[i].src = "dog" + (i + 1) + ".jpg"
}
var clockid = self.setInterval("imgloop()",1000);
function imgloop()
{
  myimg.src = imgslist[num-1].src;
  num++;
  if (num > 2) num = 1;
  form1.msg.value = num;
}
```

```
</script>
</head>
<body>
<img id="myimg" src="dog1.jpg">
<form name="form1">
  <input type="text" name="msg" size="15" />
  <input type="button" value="Stop" onclick="window.clearInterval(clockid)">
</form>
</body>
</html>
```

说明,在创建 image 对象时,使用 new Image(),其中的 Image 首字母是大写,这点特别要注意,因为 JavaScript 中的许多内置对象的首字母都是小写的,但作为类型使用时,大都是首字母大写,例如数组 Array 对象、日期 Date 对象等,新建对象时应使用首字母大写。

另外,和 Image 对象有关的事件包括两个。①onAbort 事件,当用户放弃载入一个图像时触发一个 abort 事件,于是会执行 onAbort 事件处理函数的 JavaScript 代码。例如,当在载入的过程中单击一个链接或是单击了浏览器中的“停止”按钮时,就是这种情况。②OnError 事件,当浏览器完成载入一幅图像时触发一个 load 事件。注意,是完成载入后,而不是载入开始时。

5.7.5 Link 对象与 Anchor 对象

在 HTML 文档中,有两种类型的链接,一种是在<head>…</head>之间的<link>元素,还有就是超链接标记<a>。对于<a>标记,一种形式用于定义一个超链接,即设置 href 属性,还有一种形式就是定义文档中的锚点,即只有 name 属性,没有 href 属性。

根据上述情况,document 有两个成员对象数组,即 anchors[]和 links[],anchors[]保存 Anchor 对象,links[]保存 Link 对象和超链接对象。不论是 Link 对象还是 Anchor 对象,对象的属性总是与<link>标记和<a>标记的属性相对应。

【例 5-12】 设置和更改<a>标记的相关属性。

分析: 对于超链接,使用 HTML DOM 对象,可以更改超链接的目标文件。

代码清单: links. htm

```
<html>
<body>
<form>
   <input type="button" value="Google"
         onClick="document.links[0].href='http://www.google.com'">
   <input type="button" value="百度"
         onClick="document.links[0].href='http://www.baidu.com'">
   <input type="button" value="北大天网"
         onClick="document.links[0].href='http://www.pku.edu.cn'">
</form>
<a href="javascript:alert('选择搜索引擎,然后单击下一步')">下一步</a>
<script>
document.write(document.anchors.length);
document.write(document.links.length);
```

```
</script>
</body>
</html>
```

在浏览器中打开上述页面，可以看到输出的 anchors 数组长度为 0，links 数组的长度为 1。说明 document.anchors 数组存储的只是具有 name 属性的<a>标记，如果<a>标记具有 href 属性，则对象同时被保存到 document 对象的 anchors 和 links 数组中。

下一步是网页中定义的一个超链接，存储在 document.links[0].href 中。

在<a>标记中，href 属性中应该是一个 URL，但是在这个句子中使用的是一种不常见的方式。其中，"javascript:"表示 JavaScript 语句前缀，后面为 JavaScript 语句。再如，<a href="http://www.google.com" onClick="alert('使用 Google 搜索引擎'); return (true)">，注意，最后的 return(true) 语句命名这个 URL 真正被载入，当返回 false 时是不会真正载入 http://www.google.com 这个地址的。

5.7.6　Table 对象

在 HTML 中，表格是最重要的数据组织和布局工具，利用 JavaScript 可以对表格进行插入、删除行、修改单元格内容以及显示和隐藏等各种操作。

1. 表格对象

每一个<table>标记，在内存都创建一个表格对象，其属性及方法见表 5-24。

表 5-24　Table 对象常用属性及方法

属　　性	
id	设置或返回表格的 id
className	设置或返回表格的 class 属性
width	设置或返回表格的宽度
border	设置或返回表格边框的宽度(以像素为单位)
cellSpacing	设置或返回在表格中的单元格之间的空白量(以像素为单位)
cellPadding	设置或返回单元格边框与单元格内容之间的空白量(以像素为单位)
rows[]	返回表格中所有行(TableRow 对象)的一个数组，该集合包括 <thead>、<tfoot> 和 <tbody> 中定义的所有行
cells[]	返回表格中所有单元格的一个数组。要定位一个 i 行、j 列的单元格，可写为 rows[i−1].cells[j−1]
tBodies[]	返回包含表格中所有 tbody 的一个数组
方　　法	
createCaption()	用于在表格中获取或创建 <caption> 元素
deleteCaption()	从表格删除 caption 元素及其内容
createTHead()	在表格中创建一个空的 tHead 元素
deleteTHead()	从表格中删除 tHead 元素及其内容
createTFoot()	在表格中创建一个空的 tFoot 元素
deleteTFoot()	从表格中删除 tFoot 元素及其内容

续表

方　法	
insertRow(index)	在表格 index 所在行之前插入一个新行,并返回 TableRow,表示新插入的行。若 index 等于表中的行数,则新行将被附加到表的末尾 如果表是空的,则新行将被插入到一个新的 <tbody> 段,该段自身会被插入表中
deleteRow(index)	参数 index 指定了要删除的行在表中的位置。行的编码顺序就是它们在文档源代码中出现的顺序。<thead> 和 <tfoot> 中的行与表中其他行一起编码

2. 表格行对象

表格行对象 TableRow 代表一个 HTML 表格行,在 HTML 文档中<tr>标签每出现一次,就创建一个 TableRow 对象。其属性和方法见表 5-25。

表 5-25　表格行对象 TableRow 常用属性及方法

属　性	
id	返回行的 id 属性
rowIndex	返回某一行在表格的行集合中的位置(row index)
innerHTML	设置或返回行的开始标签和结束标签之间的 HTML
cells[]	返回包含行中所有单元格的一个数组
align	设置或返回在行中的数据的水平排列方式
vAlign	设置或返回在行中的数据的垂直排列方式
方　法	
insertCell(index)	在单元格 index 之前插入一个新的单元格,并返回 TableCell 对象。若 index 等于行中的单元格数,则新单元格将被附加到行的末尾
deleteCell(index)	参数 index 指定了要删除的单元格在表中的位置

3. 单元格对象

单元格对象 TableCell 代表一个 HTML 表格单元格。在一个 HTML 文档中<td>标签每出现一次,就创建一个 TableCell 对象。其属性见表 5-26。

表 5-26　单元格对象 TableCell 常用属性

属　性	说　明
id	设置或返回单元格的 id
rowSpan	设置或返回单元格可横跨的行数
colSpan	单元格横跨的列数
width	设置或返回单元格的宽度
cellsIndex	返回单元格在某行的单元格集合中的位置
className	设置或返回元素的 class 属性
innerHTML	设置或返回行的开始标签和结束标签之间的 HTML
align	设置或返回单元格内部数据的水平排列方式
vAlign	设置或返回单元格内部数据的垂直排列方式

【例 5-13】 设计一个页面，实现表格行顺序的上下调整。

分析：在 Web 开发中，对于列表项，有时候需要调整它们的显示顺序。显示顺序的调整最简单的方法是设计一个顺序编号，在从数据库读出时，按顺序编号排序。另一种办法就是允许用户在客户端调整。

设我们设计的一个问卷项目编辑及项目顺序调整页面如图 5-12 所示。

【课程问卷管理】A(50分)二级指标管理					添加二级指标项目
序 号	二级指标	操 作		显示顺序	
1	老师讲课条理清楚，深入浅出(10)	修改	删除	上移	下移
2	老师为我们耐心答疑，并且时间充分(10)	修改	删除	上移	下移
3	我认为老师对我们和蔼可亲，平易近人(10)	修改	删除	上移	下移
4	中心网站--学习园地上的内容对我们学习有帮助(10)	修改	删除	上移	下移
5	教师教学认真，对我们要求严格(10)	修改	删除	上移	下移
单击页面右上角的“添加二级指标项目”，可以为一级指标添加二级指标项目					

图 5-12 问卷调查项目编辑界面

代码清单：surveymodal-edit2.htm

```
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312" />
<link rel = "stylesheet" type = "text/css" href = "linestable.css">
<script language = "JavaScript">
////////////////////////////////////////////////////////////////////////////////
//添加考核项目
function additem()
{
   var tempstr = "<p align = left> 二级指标: <input type = 'text' name = 'testitemname'
maxlength = '30' size = '35'>";
   tempstr += "分值: <input type = 'text' name = 'scorerate' maxlength = '2' size = '2' value = '0'>";
   tempstr += "<input type = 'button' name = 'btn1' value = '添加' onclick = 'additemsubmit()'>";
   tempstr += "<input type = 'button' name = 'btn2' value = '取消' onclick = 'modifyitemcancel()'></p>";

      editarea.innerHTML = tempstr;
}
////////////////////////////////////////////////////////////////////////////////
//上移项目
function moveup(itemnum)
{
    var rownum = parseInt(itemnum);
    var temp = table1.rows(rownum).cells(1).innerText;
    table1.rows(rownum).cells(1).innerText = table1.rows(rownum + 1).cells(1).innerText
    table1.rows(rownum + 1).cells(1).innerText = temp;

    var itemcode = table1.rows(rownum).cells(1).itemcode;
    table1.rows(rownum).cells(1).itemcode = table1.rows(rownum + 1).cells(1).itemcode;
```

```
    table1.rows(rownum + 1).cells(1).itemcode = itemcode;
}
//////////////////////////////////////////////////////////////////////////////////
//下移项目
function movedown(itemnum)
{
    var rownum = parseInt(itemnum);
    var temp = table1.rows(rownum + 2).cells(1).innerText;
    table1.rows(rownum + 2).cells(1).innerText = table1.rows(rownum + 1).cells(1).innerText
    table1.rows(rownum + 1).cells(1).innerText = temp;

    var itemcode = table1.rows(rownum + 2).cells(1).itemcode;
    table1.rows(rownum + 2).cells(1).itemcode = table1.rows(rownum + 1).cells(1).itemcode;
    table1.rows(rownum + 1).cells(1).itemcode = itemcode;
}
</script>
</head>
<body>
<table class = "table_frame" width = "70 % " cellpadding = "0" cellspacing = "0" id = "table1">
<tr height = "30">
<td class = "table_topic" colspan = "8" align = "left">
<table width = "100 % " height = "50" boder = "0"  cellpadding = "0" cellspacing = "0">
        <tr height = "10">
            <td width = "550" rowspan = "2" class = "table_topic_label"><font color = "#
FF3401">
                【课程问卷管理】</font>教书育人(50 分)二级指标管理</td>
            <td align = "center"></td>
        </tr>
        <tr height = "20">
            <td  align = "right" style = "padding - right:10px;"><a href = "#" onclick =
"additem()">添加二级指标项目</a></td>
        </tr>
        </table>
    </td>
</tr>
<tr height = "35">
    <td class = "table_head" width = "10 % ">序 号</td>
    <td class = "table_head" width = "70 % ">二级指标</td>
    <td class = "table_head" colspan = "2">显示顺序</td>
</tr>
<tr height = "30">
    <td class = "table_cell" align = "center">1 - 1</td>
     <td itemcode = "item11" class = "table_cell" align = "left" onclick = "alert(this.
itemcode)">人品高尚,有良好的社会责任感,有爱心和同情心(30 分)</td>
    <td class = "table_cell" align = "center"><a href = "#" onclick = "moveup('1')">上移</a>
</td>
    <td class = "table_cell" align = "center"><a href = "#" onclick = "movedown('1')">下移</a>
</td>
</tr>
<tr height = "30">
    <td class = "table_cell" align = "center">1 - 2</td>
```

```
    < td itemcode = "item12" class = "table_cell" align = "left" onclick = "alert(this.
itemcode)">具有良好的敬业精神(10 分)</td>
    <td class = "table_cell" align = "center"><a href = "#" onclick = "moveup('2')">上移</a>
</td>
    <td class = "table_cell" align = "center"><a href = "#" onclick = "movedown('2')">下移</a>
</td>
</tr>
<tr height = 60>
    <form name = "form1" method = "post" action = "surveymodal - edit2save.jsp" target = "_
self">
    <input type = "hidden" name = "head2list" value = "<% = head2list %>">
    <td id = "editarea" style = "background - color: #F7FBFF;" colspan = "8">
单击页面右上角的"添加二级指标项目",可以为一级指标添加二级指标项目.
    </td>
    </form>
</tr>
</table>
</body>
</html>
```

上述代码演示了客户端强大的程序功能,这可以减少和 Web 服务器的连接次数,减少服务器的负载,提高整个系统的性能。上述示例程序,有两处核心代码。

(1) 在表格第 2 列中,我们在<td>标记中使用了 itemcode 属性的关于标记的属性,这是一种全新的用法,因为在 HTML 规范中,<td>标记中无此属性。因为对于许多标记,我们不仅需要标记的内容(即 innerText 和 innerHTML),有时候还需要更多的信息。

从 HTML DOM 对象的思想出发,对于 HTML 中的标记,除了规范给定的属性外,用户还可以根据需要设定标记属性,JavaScript 都将其转化为 DOM 对象的成员属性,这就大大提高了标记和 HTML DOM 的灵活性。这种用法,还常常用于复选框的分类等。

(2) 对于添加项目函数 additem(),示例代码没有给出其中涉及的所有函数,但它演示了 Web 开发中数据维护的三个主要问题:添加、修改和删除,图 5-12 给出了一个很好的功能组织界面,界面友好始终是 Web 系统设计和开发的重要内容。

5.7.7 Form 对象

在 Web 页中,表单(form)是人机交互的主要手段。每一个<form>标记都在内存中创建一个 Form 对象,Form 对象封装了 HTML 的<form>标记。Form 对象属性及方法见表 5-27。

表 5-27 Form 对象常用属性及方法

一 般 属 性	
id	返回 form 标记的 id 属性
name	设置或返回表单的名称
method	设置或返回将数据发送到服务器的 HTTP 方法
enctype	设置或返回表单用来编码内容的 MIME 类型

续表

一 般 属 性	
action	设置或返回表单的 action 属性
target	设置 action 属性指定服务器程序输出结果要显示的帧或窗口名称
length	返回表单中的元素数目
elements[]	包含表单中所有元素的数组
事 件 属 性	
onsubmit	当用户单击表单中的 Submit 按钮而提交一个表单时执行该方法。由于用户输入的有效性验证问题,在表单中,通常不使用提交(Submit)按钮,而使用普通的按钮(button),在按钮的 onclick 事件属性中首先进行有效性验证,最后执行 document. form-name. submit()提交表单
onreset	在重置表单元素之前调用
方　　法	
reset()	把表单的所有输入元素重置为它们的默认值
submit()	提交表单,调用 form 的 onsubmit()方法。如果希望使用普通按钮的提交表单,则不要在<form>中设置 onsubmit 事件属性

在<form>中通常包含了一系列的输入元素,它们也被创建为响应的对象,并被保存到 Form 对象的 elements[]中。对于输入对象的属性和方法,和输入标记的属性对应,在此不再详细介绍。

1. 访问 Form 对象

在一个 HTML 文档中,可以包含多个<form>,因此在 JavaScript 中,访问 Form 对象有多种实现方法。

方法一:通过 form 名称访问。

在<form>标记中,包含一个 name 属性,它对应 Form 对象的对象名,JavaScript 程序可以通过表单名来访问表单对象,即:

```
obj = document.form - name
```

当然也可以利用 document. getElementsByName("form-name")来返回 Form 对象。

方法二:通过 document 对象 forms[]属性。

除了使用 form 对象名来访问页面中的表单外,还可以通过 document 对象的 forms 对象数组来访问 Form 对象,有两种用法:document. forms[num],num 为 0,1,…,或者 document. forms["form-name"],都返回 Form 对象。需要说明的是,在 JavaScript 中要对 form 引用的条件是:必须先在页面中用<form>标记创建表单,并将定义表单部分放在引用之前。

2. 访问 form 中的元素

一个表单是由若干的表单控件组成的,这些控件包括文本框(text)、单选按钮(radio)、复选框(checkbox)、按钮(button)等。每一个控件,在内存中都创建相应的内存对象。在 JavaScript 中要访问这些基本元素,必须通过对应特定的 form 元素的数组下标或 form 元

素名来实现。每一个元素主要是通过该元素的属性或方法来引用。其引用表单元素对象，有多种方法。

方法一： 通过 Form 对象的 elements[]数组。

```
Obj = document.form - name.elements["element - name"]
```

方法二： 直接通过输入元素名。

```
Obj = document.form - name.element - name
```

两种方法都返回表单输入元素对应的 DOM 对象。不同的 form 控件，对应的内存对象的属性和方法也不相同，但每一个内存对象包含的属性和方法均与其对应的 HTML 标记对应。例如，对于 text 内存对象，对应一个单行文本框输入控件。在 HTML 中，标记单行文本框 text 输入的一般形式是：

```
<input type = "text" name = "input - name" value = "default - value">
```

因此，对应于每一个 text 对象，其包含的属性包括 name 属性、value 属性等。对对象属性的操作，即通过对象的点记法来完成，即：对象名.[属性|方法]。

对于每一个输入对象，通常还包含一组通用的方法，如：blur()方法，将当前焦点移到后台；select()方法，选择 text 框内的文本。包含的事件有：onFocus，当 text 获得焦点时，产生该事件；onBlur，从元素失去焦点时，产生该事件；onSelect，当文字被选中，产生该事件；onChange，当元素值改变时，产生该事件。

【例 5-14】 设计一个用户信息输入页面，完成用户个人信息的填写，当用户提交表单时，显示用户的输入信息。

分析： 在上网过程中，数据输入大都是通过表单来完成的，如何提取表单中的用户输入数据，包括文本框、复选框、单选按钮、列表框、多行文本框等输入数据，以及对输入进行控制和有效性验证是 Web 编程中的共性和基础性问题。

用户信息输入界面设计如图 5-13 所示。

图 5-13　表单输入界面

上述表单包括主要的输入元素，除了要获取输入元素的数值外，经常还需要对用户输入进行控制，例如：检查文本框输入数据的合理性，设置文本框的只读属性，限制文本输入的字符个数，等等。下面的代码是上述功能的实现。

代码清单： personinfo.htm

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;charset=gb2312">
<style type="text/css">
body{margin-top:10px;background-color:#FFF8EB;font-size:30px;}
td{font-size:13px}
</style>
<script type="text/javascript">
/////////////////////////////////////////////////////////////////////////////////////
//用户名为只读属性
function checkname(obj)
{
  if (obj.readOnly) alert("用户名文本框为只读属性!");
}
/////////////////////////////////////////////////////////////////////////////////////
//返回单选按钮的值
function getsexval()
{
  var sexValue = "";
  for (i = 0;i < form1.sex.length;i++)
  {
      if (form1.sex[i].checked)
      {
          sexValue = form1.sex[i].value;
          break;
      }
   }
  return sexValue;
}
/////////////////////////////////////////////////////////////////////////////////////
//获取复选框选择,多个复选框元素可以设置相同的 name 属性
function getcheckboxval()
{
    var likestr = "";
    var length = document.form1.mylike.length;
    for(i = 0;i < length;i++)
    {
        if(document.form1.mylike[i].checked)
          likestr  += document.form1.mylike[i].value + ";";
    }
    return likestr.substring(0,likestr.length - 1);
}
/////////////////////////////////////////////////////////////////////////////////////
//获取下拉列表输入,取<optin value=></option>中 value 的值,或标记的文本
function getlistboxval(obj,str)
```

```
{
   if (str == "value")
       return obj.value;
   else
       return obj.options[obj.selectedIndex].text;
}
///////////////////////////////////////////////////////////////////////////
//限制多行文本域输入的字符个数
var LastCount = 0;
function CountStrByte(Message,Total,Used,Remain)
{
  var str     = Message.value;
  var length = Message.value.length;
  var maxnum = Total.value;
  var ByteCount = 0;
  if(LastCount != length)
  {
   for (i = 0;i < length;i++){
       ByteCount   = (str.charCodeAt(i)<= 256) ? ByteCount + 1 : ByteCount + 2;
       if (ByteCount > maxnum) {
           Message.value = str.substring(0,i);
           alert("个人简介最多不超过" + maxnum + "个字节!一个汉字为两字节.");
           ByteCount = maxnum;
           break;
       }
   }
   Used.value = ByteCount;
   Remain.value = maxnum - ByteCount;
   LastCount = length;
  }
}
///////////////////////////////////////////////////////////////////////////
//验证身份证输入位数是否正确
function checkpcard(str)
{
   var Expression = /\d{15}|\d{18}/;
   var objExp = new RegExp(Expression);
   if (objExp.test(str) == true) return true;
   else
   {
       alert("身份证位数不对!");
       return false;
   }
}
///////////////////////////////////////////////////////////////////////////
//验证邮箱地址是否正确
function checkemail(str)
{
   //在 JavaScript 中,正则表达式只能使用"/"开头和结束,不能使用双引号
   var Expression = /\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;
   var objExp = new RegExp(Expression);
```

```
        if (objExp.test(str) == true) return true;
        else
        {
            alert("用户邮箱格式不正确!");
            return false;
        }
    }
    ////////////////////////////////////////////////////////////////////////////////////
    //数据有效性验证
    //提交用户输入信息
    function form1submit()
    {
       if (escape(document.form1.useraccount.value).indexOf("%u")! = -1)
       {
          alert("用户账户不能包含汉字,请重新输入");
          document.form1.useraccount.focus();
          return false;
       }
        // 6~20 位的字母、数字、下划线、句点组成的密码
        var Expression = /^[A-Za-z0-9]{1}([A-Za-z0-9]|[._@]){5,19}$/;
        var objExp = new RegExp(Expression);
        if (objExp.test(form1.password.value) == false)
        {
            alert("密码由 6~20 位的字母、数字、点、下划线、@组成且首字符为字母");
            form1.password.focus();
            return false;
        }
        if (!checkpcard(document.form1.pcard.value))
        {
            document.form1.pcard.focus();
            return false;
        }
        if (document.form1.email.value == "")
        {
             alert("必须填写用户邮箱!");
             document.form1.email.focus();
             return false;
        }
        if (!checkemail(document.form1.email.value))
        {
            document.form1.email.focus();
            return false;
        }
        var str = "个人基本信息:\n";
        if (document.form1.myname.value ! = "")
            str += "姓名:" + document.form1.myname.value + "\n";
        str += "性别:" + getsexval() + "\n";
        str += "教育状况:" + getlistboxval(form1.grade,'text') + "\n";
        str += "个人兴趣:" + getcheckboxval() + "\n";
        if (document.form1.mybrief.value ! = "")
            str += "个人简介:" + document.form1.mybrief.value + "\n";
```

```
    document.form1.totalinfo.value = str;
    alert(document.form1.totalinfo.value);
    form1.submit();
}
</script>
</head>
<body>
<table width = "500" height = "100 %" cellspacing = "0" border = "0" align = "center">
<tr height = "100 %">
<td align = "center">
<form name = "form1" method = "post" action = "personeditsave.jsp" target = "_self">
<table width = "100 %" border = "0" cellspacing = "0" cellpadding = "0" style = "border: 1px
solid red;padding - left:10px">
<tr height = "35">
     <td colspan = "2"><img src = "square.gif" valign = "absmiddle">用户个人资料</td>
</tr>
<tr>
     <td height = "1" colspan = "2"><img src = "line2.gif" height = "1"></td>
</tr>
<tr height = "30">
   <td align = "right">用户账户:</td>
   <td>
      <input type = "text" name = "useraccount" value = "haoxw365" size = "20" disabled>(用户
账户不能自行修改)
   </td>
</tr>
<tr height = "30">
   <td width = "100" align = "right">姓名:</td>
   <td>
       <input type = "text" name = "myname" value = "Hao" readOnly = "true"
            onKeyDown = "checkname(this)"> 性别
       <input type = "radio" name = "sex" value = "male" checked>男
       <input type = "radio" name = "sex" value = "female">女
   </td>
</tr>
<tr>
     <td align = "right">教育状况:</td>
     <td align = "left">
       <select name = "grade">
           <option value = "博士">博士</option>
           <option value = "硕士">硕士</option>
           <option value = "本科" selected>本科</option>
       </select>
     </td>
</tr>
<tr height = "30">
   <td align = "right">身份证号码:</td>
   <td>
      <input type = "text" name = "pcard" maxlength = "18" size = "45">
   </td>
</tr>
```

```
<tr height="30">
   <td align="right">个人邮箱:</td>
   <td>
       <input type="text" name="email" size="45">
   </td>
</tr>
<tr>
     <td align="right">个人兴趣:</td>
     <td> 
         <input type="checkbox" name="mylike" value="体育">体育
         <input type="checkbox" name="mylike" value="音乐">音乐
         <input type="checkbox" name="mylike" value="旅游">旅游
     </td>
</tr>
<tr height="30">
   <td align="right">自我介绍:</td>
   <td>
         <textarea name="mybrief" cols="45" rows="6"
         onKeyDown="CountStrByte(this.form.mybrief,this.form.total,this.form.used,this.
         form.remain);"
         onKeyUp="CountStrByte(this.form.mybrief,this.form.total,this.form.used,this.
         form.remain);">
         </textarea>
     </td>
</tr>
<tr>
     <td colspan="2" align="center">最多允许
       <input type="text" name="total" value="100" size="3" disabled>个字节 已用字节:
       <input type="text" name="used" value="0" size="3" disabled>剩余字节:
       <input type="text" name="remain" value="100" size="3" disabled>
     </td>
</tr>
<tr height="30">
     <td colspan="2" align="center">
       <input type="button" name="myok" value="确定" onclick="form1submit();"> 
       <input type="reset" name="mycan" value="取消">
     </td>
</tr>
</table>
<input type="hidden" name="useraccount" value="sdu12345">
<input type="hidden" name="password" value="111111">
<input type="hidden" name="totalinfo" value="">
</form>
</td>
</tr>
</table>
</body>
</html>
```

对于上述代码,说明如下。

(1) 在文档的头部,我们定义了嵌入式样式表,在<body>样式定义中,只对<body>

内的<p>等有影响,其 font-size 不影响表格中的文字,要设置表格中文字的大小,只能在<td>标记中设置,即使在<table>中设置同样无效。

(2) 关于表格线,在<table>中设置 border=1,则表格的表格、单元格都含有表格线,设置<table>的 style="border:1px solid red",则只有表格四周的边线。单元格无表格线。

(3) 表单提交按钮设计为 Button 类型,而不是 Submit 类型,主要是增加可读性,在 form1submit()中进行数据的有效性验证。

(4) 对于 JavaScript 函数,参数基本上有两类,一类是字符串,另一类是对象。例如,checkemail(),如果参数是对象,则调用时可使用 checkemail(this),在函数内部很容易取对象的 value 值,如果参数是字符串,实际参数就写为 this. value,即当前文本框的 value 值。

(5) 表单中使用了 hidden 输入,主要用于客户端和服务端之间的数据交换,如果是服务器页(如 jsp),很容易把服务端的数据通过 hidden 传递,例如:<input type=hidden name="nowdate" value="<%=new Date()%>",这样客户端就得到了服务器上的时钟。

5.7.8 Event 对象

在网页浏览中,会有各种各样的鼠标和键盘操作,这些事件的发生,都在内存中创建 Event 对象,Event 对象封装了键盘按键的状态、鼠标指针的位置、鼠标按键的状态等信息。

1. Event 对象常用属性

Event 对象属性见表 5-28。

表 5-28 Event 对象常用属性

属　　性	说　　明
clientX、clientY	返回当事件被触发时,鼠标指针的 X、Y 坐标
screenX、screenY	返回当事件被触发时,相对于屏幕的横坐标和列坐标
button	返回一个整数,指示当事件被触发时哪个鼠标按键被按下,0、1、2 分别代表鼠标左键、中键和右键
ctrlKey	返回一个布尔值,指示当事件发生时,Ctrl 键是否被按下并保持住
altKey	返回一个布尔值,指示在指定的事件发生时,Alt 键是否被按下并保持住
shiftKey	返回一个布尔值,指示当事件发生时,Shift 键是否被按下并保持住
relatedTarget	返回与事件的目标节点相关的节点。对于 mouseover 事件来说,该属性是鼠标指针移到目标节点上时所离开的那个节点。对于 mouseout 事件来说,该属性是离开目标时,鼠标指针进入的节点。对于其他类型的事件来说,这个属性没有用
type	返回当前 Event 对象表示的事件的名称
target	返回事件的目标节点对象(触发该事件的节点),如生成事件的元素、文档或窗口
keyCode	对于 keypress 事件,该属性声明了被敲击的键生成的 Unicode 字符码。对于 keydown 和 keyup 事件,它指定了被敲击的键的虚拟键盘码
cancelBubble	如果事件句柄想阻止事件传播到包容对象,必须把该属性设为 true

2. 可能的事件

在浏览器中可能的事件主要有：onclick（鼠标单击）、ondblclick（鼠标双击）、onmousedown(按下鼠标按键)、onmousemove(鼠标移动)、onmouseout(鼠标指针从某元素移开)、onmouseover(鼠标指针移到某元素之上)、onmouseup(松开鼠标按键)、onkeydown(按下键盘按键)、onkeypress(按下并松开键盘按键)、onkeyup(松开键盘按键)、onfocus(元素获得焦点)、onblur(元素失去焦点)、onchange(改变域的内容)、onselect(选中文本)、onload(网页或一幅图像完成加载)、onunload(退出页面)、onabort(图像加载中断)、onerror(加载文档或图像时出错)、onresize(调整窗口或框架大小)、onreset(单击“重置”按钮)、onsubmit(单击“提交”按钮)等。

【例 5-15】 编写输入元素事件处理函数，当按 Enter 键时，将输入焦点自动移动到下一个输入元素。

分析：在使用表单输入时，默认情况下，按 Enter 键，则提交表单。这可能不符合我们的交互习惯，我们的一般习惯是，按 Enter 键后，使输入焦点移向下一个表单元素，而不是提交表单。

代码清单：

```
<script type="text/javascript">
function myenter(nextobj)
{
  if (event.keyCode == 13)
    nextobj.focus();
}
</script>
```

然后在每一个 Input 控件中，可以修改 onkeypress 事件属性为 onkeypress ="myenter(下一个输入元素)"。例如：

```
<form name="form1">
<input type="text" name="useraccount" onkeypress="myenter(form1.password)" />
<input type="text" name="password" onkeypress="myenter(form1.username)" />
</form>
```

则用户输入完用户账户后，按 Enter 键，输入焦点将转到 name="password"的输入框。

5.7.9 应用举例

在 Web 系统的开发中，表格和表单是使用最多的页面元素，利用 JavaScript 可以动态地在表格中插入、删除表格行，同时也可以隐藏表格行和单元格。结合表单输入，通过表格来显示，实现一种类似数据库记录的输入界面，最后一次性提交到 Web 服务器。

在我们进行 GSL 系统的研发中，设计了考试模型的编辑界面，对于每一类答题，列出题库中的题目，选择需要的题目，并给定分值，最后提交。界面如图 5-14 所示。

分析：采用<iframe>来显示下部的列表，正常运行时，列表来自数据库查询所得数据。在列表项右侧，单击“添加”按钮，则相应的项目添加到上部的已选题目列表，对已选题目列



图 5-14 考试模型编辑设计页面

表中的题目，如果要放弃选择，单击右侧的“取消选定”按钮即可。最后单击“确定”按钮，将所选题目列表发送到服务器，更新数据库中的课程考试模型。

界面设计两个 HTML 文件，一个为主体页面对应的 html 文件，文件名为 tableformmain.htm，另一个为<iframe>中显示的文件，文件名为 tableformlist.htm。

（1）tableformmain.htm 代码清单如下：

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<link rel="stylesheet" type="text/css" href="frametable.css">
<link rel="stylesheet" type="text/css" href="common.css">
<script>
///////////////////////////////////////////////////////////////////////////////////
//动态在表格中添加一行
//子窗口调用的函数，必须单独写在一个独立的脚本定义段内，否则，出错
function addonerow(str)
{
   if (table1.rows(1).id == "row0")
   {
     //alert("第一次添加..."),提示信息占用了一行,需要隐藏掉,以免存入数据库
     document.all.item('row0').style.display = "none"
   }
   var s = new Array(2);
   var ch1 = "\20";
   s = str.split(ch1);
   var rowsnum  =  document.all.table1.rows.length;
   var newRow,newCell;
   //在最后一行的前面插入一行,每行 4 个单元格,并设置行的 id 属性
   var newRow = table1.insertRow(rowsnum - 1);
   newRow.id = "row" +  rowsnum;
```

```
for(var j = 0;j < 4;j++)
{
    newCell = newRow.insertCell(j);
}
// IE 浏览器动态设置样式不能直接给行或列指定 class 属性
//要先把样式放到一个对象的 attribute 中,然后再把这个对象设置到行或列中
//第 1 列
var objStyle = newRow.cells(0).getAttribute("style");
if (objStyle == "[object]")
{
    objStyle.setAttribute("cssText","font - size:12px;color: # 333333;background - color:
    # b3cfe4;");
    newRow.cells(0).setAttribute("style",objStyle);

 }
else
{
    newRow.cells(0).setAttribute("class","textrow");
 }
//第 2 列
objStyle = newRow.cells(1).getAttribute("style");
if (objStyle == "[object]")
{
    objStyle.setAttribute("cssText","font - size:12px;color: # 333333;background - color:
    # b3cfe4;");
    newRow.cells(1).setAttribute("style",objStyle);
 }
else
{
    newRow.cells(1).setAttribute("class","textrow");
 }
//第 3 列
objStyle = newRow.cells(2).getAttribute("style");
if (objStyle == "[object]")
{
    objStyle.setAttribute("cssText","font - size:12px;color: # 333333;background - color:
    # b3cfe4;");
    newRow.cells(2).setAttribute("style",objStyle);
 }
else
{
    newRow.cells(2).setAttribute("class","textrow");
 }
//第 4 列
objStyle = newRow.cells(3).getAttribute("style");
if (objStyle == "[object]")
{
    objStyle.setAttribute("cssText","font - size:12px;color: # 333333;background - color:
    # b3cfe4;");
    newRow.cells(3).setAttribute("style",objStyle);
 }
```

```
    else
    {
        newRow.cells(3).setAttribute("class","textrow");
     }
    //为新添加的行的各个单元格赋值
    newRow.cells(0).innerText = s[0];
    newRow.cells(0).align = "center";
    newRow.cells(0).width = "15 % ";

    newRow.cells(1).innerText = s[1];
    newRow.cells(1).align = "left";
    newRow.cells(1).width = "60 % ";

    newRow.cells(2).innerHTML = "分数: < input type = 'text' name = 'score' maxlength = '3' size =
'2' value = '0' onblur = \"checkscore(this)\">";
    newRow.cells(2).align = "center";
    newRow.cells(2).width = "10 % ";

    newRow.cells(3).innerHTML = "< input type = 'button' name = 'btn' value = '取消选定' onclick =
\"cancelselect('row" + rowsnum + "')\">";
    newRow.cells(3).align = "center";

    //newRow.height = "30";
    window.scrollTo(0,document.body.scrollHeight);
}
</script >

< script >
/////////////////////////////////////////////////////////////////////////////////
//检查输入的分数是否合适
function checkscore(obj)
{
    var scorestr = obj.value;
    var Expression = /^\d{1,3} $ /;
    var objExp = new RegExp(Expression);
    if (objExp.test(scorestr) == false)
    {
        alert("数字为不大于 100 的整数!");
        thisobj.focus();
        return false;
    }
    return true;
}
/////////////////////////////////////////////////////////////////////////////////
//隐藏一个 id 所标记的区域,如 div、表格行、单元格等
//当一个 id 标记的对象隐藏显示后,其值不可访问,出现 undefined 提示
function cancelselect(area)
{
   if (document.all.item(area).style.display == "none")
      document.all.item(area).style.display = "block";
   else
```

```
        document.all.item(area).style.display = "none";
    }

    ////////////////////////////////////////////////////////////////////////////////////
    //将用户选择的题目,形成 testcodelist 列表传回服务器保存到数据库
    //
    function form1submit()
    {
       //将表格中的内容形成一个考试题串,题目之间用"\20"分开
       var tempstr = "",s1,s2,s3;
       var rowsnum,i = 0,j = 1,k;
       rowsnum = document.all.table1.rows.length;

       //只有三行,且第一行是提示行,则表明无数据
       if (rowsnum == 3 && table1.rows(1).id == "row0")
          return false;

       //第一行、第二行(隐藏)、最后一行不是题目数据
       var totalscore = 0;
       for (i = 1;i < rowsnum - 1;i++)
       {
           if (table1.rows(i).style.display! = "none")
           {
               s1 = table1.rows(i).cells(0).innerText;
               s2 = table1.rows(i).cells(1).innerText;
               //只有一条记录时,score 未形成数组,分两种情况
               if (rowsnum == 3 && table1.rows(1).id! = "row0" || rowsnum == 4 && table1.rows(1).id
    == "row0")
                   s3 = form1.score.value;
               else
                   if (table1.rows(1).id == "row0")
                       s3 = form1.score[i - 2].value;
                   else
                       s3 = form1.score[i - 1].value;
               tempstr += s1 + "," + s2 + "," + s3 + "\20";
               totalscore += parseInt(s3,10);
            }
       }
       //原先有数据,但取消了所有的行,即指定的题目列表变为空
       if (tempstr! = "" && totalscore! = 100)
       {
          alert("已选题目总分为: " + totalscore + ",总分应为 100 分")
          return false;
       }
       form1.testitemlistnew.value = tempstr;
       form1.submit();
    }
    </script>
    </head>

    <body style = "margin - top:0px;">
```

```
<form name="form1" action="tableformmainsave.jsp">
<input type="hidden" name="oneteststr" value="">
<input type="hidden" name="testitemlistnew" value="">

<table id="table1" width="100%" class="info-table" cellspacing="0">
<tr height="35">
    <td class="titlerow" colspan="4" align="left">已选题目</td>
</tr>
<tr height="35" id="row0">
    <td class="table_head" width="15%">题目代码</td>
    <td class="table_head" width="60%">题目名称</td>
    <td class="table_head" width="10%">分值比例</td>
    <td class="table_head">操作</td>
</tr>
<tr height="30">
  <td class="bottomrow" colspan="4" align="right">
      <input type="button" name="btn2" value="确定" onclick="form1submit()">
      <input type="button" name="btn2" value="取消">
  </td>
</tr>
</table>
</form>

<table id="table2" width="100%" class="info-table" cellspacing="0">
<tr>
  <td colspan="3">
    <iframe name="testlistframe" src="tableformlist.htm" width="100%" height="330"
frameborder="0" scrolling="yes">
    </iframe>
  </td>
</tr>
</table>
</body>
</html>
```

(2) <iframe>内显示的文件 tableformlist.htm 的代码清单如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<link rel="stylesheet" type="text/css" href="linestable.css">
<link rel="stylesheet" type="text/css" href="common.css">
<script>
////////////////////////////////////////////////////////////////////////////////////
//实现 iframe 内部页面直接调用该 iframe 所属父窗口自定义函数的方法
//比如有 A 窗口,A 内有个 iframe B,B 中装载的是 C 页面
function toaddonerow(itemnum)
{
    //将题目串付给父窗口的隐藏变量
    var str;
    var ch1="\20";
```

```
        str = eval("item" + itemnum + "1.innerText") + ch1;
        str += eval("item" + itemnum + "2.innerText");

        //调用父窗口函数,在已选列表中新添加的题目显示
        window.parent.form1.oneteststr.value = str;
        window.parent.addonerow(str);
    }
    </script>
    </head>

    <body>
    <table class = "table_frame" width = "100 % " cellpadding = "0" cellspacing = "0">
    <tr height = "35">
        <td class = "table_head" width = "15 % ">题目代码</td>
        <td class = "table_head" width = "75 % ">题目名称</td>
        <td class = "table_head">操 作</td>
    </tr>
    <tr height = "30">
        <td id = "item11" class = "table_cell" align = "center"> 201107001 </td>
        <td id = "item12" class = "table_cell" align = "left"> Line1 …</td>
        <td class = "table_cell" align = "center"><a href = " # " onclick = "toaddonerow('1')">添加
    </a></td>
    </tr>
    <tr height = "30">
        <td id = "item21" class = "table_cell" align = "center"> 201107002 </td>
        <td id = "item22" class = "table_cell" align = "left"> Line2 …</td>
        <td class = "table_cell" align = "center"><a href = " # " onclick = "toaddonerow('2')">添加
    </a></td>
    </tr>
    </table>
    </body>
    </html>
```

上述页面较好地使用了表单、表格操作,同时用到了<iframe>等技术,在 Web 开发中具有较好的参考价值。主要技术点有如下几个。

(1) 一个页面中调用另一个页面的函数的方法,以及为另一个页面的变量赋值。

(2) 对于<form>中输入标记的 name 重名问题。在 JavaScript 中,允许多个输入域具有相同的 name 值。例如:本题中的分数,如果有两行或以上,则有多个 name="score"的输入文本框,对于多个 score,JavaScript 将创建一个数组对象来存储,数字名为 score。但是,在系统运行时,如果是一行,则只能创建一个简单的 DOM 对象,而不是数组。这是在程序调试时最容易遇到的问题。

(3) 动态的表格处理,单元格样式设置是一个难点,各个列的样式类似,但必须分别设置。

5.8 使用 AJAX 技术

在 Web 应用蓬勃发展的今天,新的技术不断出现。在客户端编程方面,继 DOM 之后,为了更新页面局部的需要,出现了 AJAX 技术,它开创了一种新的 JavaScript 编程的思路和

方法。AJAX技术一旦出现，就受到了开发人员的欢迎，并得到了迅速的发展，并被广泛应用于许多需要实时刷新的页面中。

5.8.1 AJAX基础

AJAX技术是由Jesse James Garrett于2005年2月在一篇文章中提出来的，是Asynchronous JavaScript and XML(异步JavaScript和XML)的缩写，AJAX提供与服务器异步通信的能力，一个最简单的应用是无须刷新整个页面而在网页中更新一部分数据。当时，AJAX只是一种设想，其灵感来源于需要用Flash来实现一些网络应用的功能，但是，由于对Flash的不熟悉，尝试用传统的Web技术来达到Flash的效果，即用JavaScript和XML这两种传统的Web技术来实现，这就出现了AJAX的概念。现在，这个术语已经用来泛指所有允许浏览器在不刷新整个页面的情况下与服务器通信的技术。

什么是AJAX技术呢？AJAX技术实际上是把JavaScript、CSS、DOM和HTML结合起来的一种新用法。这种结合并不是新概念，在动态HTML中，人们已经将这些技术结合在一起使用了，AJAX技术的独到之处在于它在服务器端使用了异步(Asynchronous)处理技术。

Web应用软件在运行时需要大量的页面，用户在网页上输入数据时，单击“提交”按钮，客户端浏览器则把这些信息发送到服务器端，服务器根据用户的操作发送一个新页面到客户端。例如，在一个登录页面中，除了登录表单，页面上通常还包含网站品牌信息、导航条和版权声明等。对于传统的登录页面，当用户提交表单后，服务器将比较用户在表单中输入的数据与数据库中保存的登录信息是否一致。如果用户输入的数据不正确，服务器就把与原来相同的登录页面重发给用户，而这个页面与原来的页面相比可能只是多了“登录失败”的消息。为了这小小的改变必须重新传输整个网页。用户每发出一个请求，整个网页就要被刷新一次，即页面的加载与用户的请求是同步的。

刷新整个页面除了带来较大的网络流量外，对于一些聊天类的网站，频繁的页面刷新必然会产生闪烁，影响用户的视觉体验。当采用AJAX技术后，情况将会大大改善，当用户提交表单后，如果登录失败，将不再刷新整个网页，而是仅仅在页面上增加了“登录失败”的消息文本，即AJAX技术可以实现网页的局部刷新，而页面上的所有没有被刷新的信息都是提交表单前页面的内容。这无论是对于服务器的CPU开销还是对网络的传输开销，无疑都减轻了不少压力，也避免了页面闪烁现象的发生。

总之，AJAX是一种客户端实现方法，开发人员不需要学习一种新的语言，在大多数现代浏览器中都能使用。AJAX不关心服务器是什么，使用AJAX技术，不必丢掉原先掌握的服务器端技术，它可以与ASP、JSP、PHP等服务端脚本交互。尽管存在一些很小的安全限制，网站设计者还是可以充分利用原有的知识，很容易地使用AJAX技术。

5.8.2 XMLHttpRequest对象

AJAX技术的组成元素涉及JavaScript、CSS、DOM、XML HTTP和XML等内容，对于JavaScript、CSS、DOM和XML，在前面的章节中都已经做了较全面的介绍，本节将详细介绍XML HTTP相关对象的概念及使用方法。

1. XMLHttpRequest 对象

1999 年春,在 IE 5.0 中,增加了一个新的 ActiveX 控件,即 XMLHttpRequest 对象(XHR),当时,这个对象主要是在 IE 中使用。从 Mozilla 1.0 和 Safari 1.2 开始,对 XHR 对象的支持开始普及。这个很少使用的对象和相关的基本概念甚至已经出现在 W3C 标准中,即 DOM Level 3 加载和保存规约(DOM Level 3 Load and Save Specification)。现在,特别是随着 Google Maps、Google Suggest、Gmail、Flickr、Netflix、A9 等应用变得越来越炙手可热,XHR 已经成为事实上的标准,是 AJAX 技术的核心组成部分。

XMLHttpRequest 对象位于客户端浏览器中,是用来实现网页与 Web 服务器之间异步通信的对象,通过它可以在不进行整个网页刷新的情况下向服务器发出请求、接收响应等工作。AJAX 技术工作机制如图 5-15 所示。

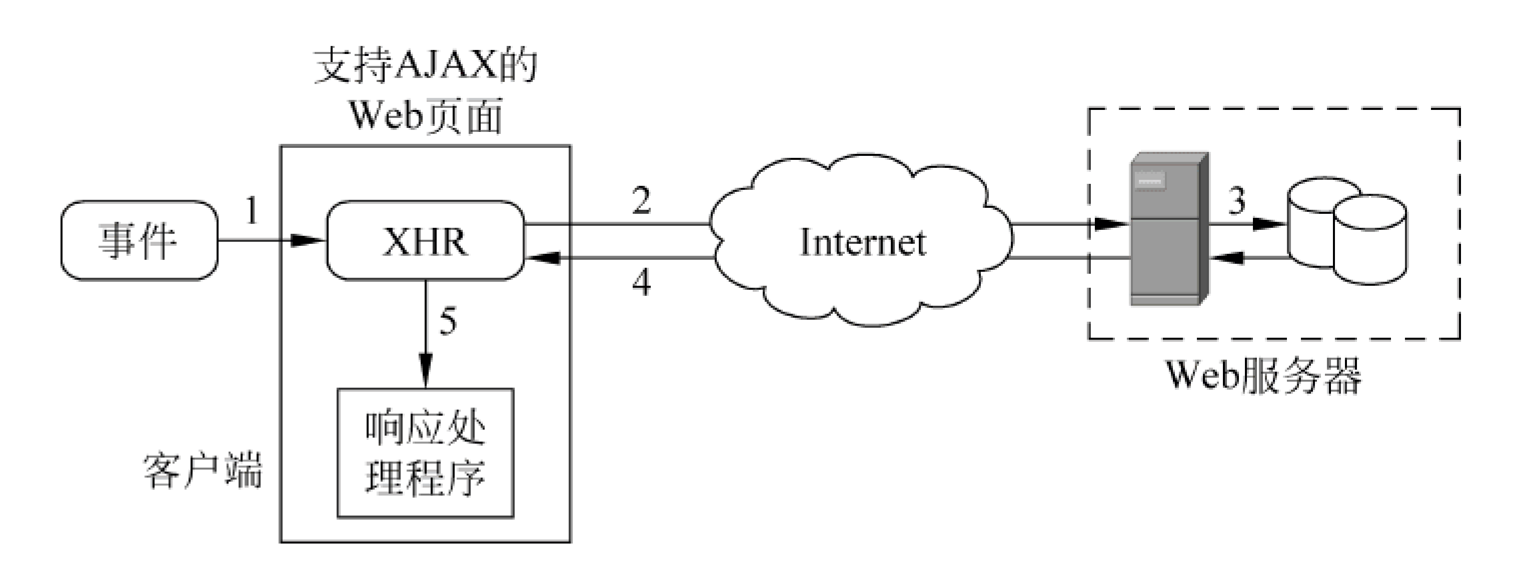


图 5-15 AJAX 技术工作机制

对于一个支持 AJAX 的 Web 面来说,与服务器进行异步数据通信的过程如下。

(1) 当要求进行与服务器异步通信的某一事件发生时,事件处理程序将调用 XHR,设置该对象相关的属性参数。

(2) 由 XHR 向服务器发出请求,请求通过 Internet 发送到 Web 服务器。当服务器收到请求后,运行指定的服务器端程序。

(3) 服务器端程序执行中如果包含数据库访问的命令,则连接数据库服务器完成数据库的相关操作,结果返回到 Web 服务器。

(4) Web 服务器将响应信息通过 Internet 发回到客户端,浏览器将这些响应信息交给 Web 页面的 XHR 对象。

(5) 启动相应的处理程序,通过该文档的 DOM 模型完成页面的更新工作。

AJAX 技术等于在客户端页面和服务器之间安插了一个中转站,JavaScript 脚本先把请求从页面发送给这个中转站,再由这个中转站把请求发给服务器;服务器对请求处理后,先把响应发给中转站,再由中转站将响应转发给页面,客户端页面的脚本程序就可以根据收到的数据进行网页的更新工作。这个中转站的角色就由客户端的 XMLHttpRequest 对象承担。在客户端的响应数据处理程序一般为 XHR 对象状态改变的事件处理函数,具体的网页更新工作可通过当前页面的 DOM 模型完成。

2. 创建 XMLHttpRequest 对象

在使用 XMLHttpRequest 对象之前,必须先创建一个 XMLHttpRequest 对象。由于

XMLHttpRequest 不是一个 W3C 标准，因此需要针对不同的浏览器采用不同方法创建 XMLHttpRequest 的实例。Internet Explorer 把 XMLHttpRequest 实现为一个 ActiveX 对象，其他浏览器(如 Firefox、Safari 和 Opera)把它实现为一个本地 JavaScript 对象。由于存在这些差别，JavaScript 代码中必须包含有关的逻辑，从而使用 ActiveX 技术或者使用本地 JavaScript 对象技术来创建 XMLHttpRequest 的一个实例。

在这里为了明确该如何创建 XMLHttpRequest 对象的实例，并不需要那么详细地编写代码来区别浏览器类型，只是检查浏览器是否提供对 ActiveX 对象的支持。如果浏览器支持 ActiveX 对象，就可以使用 ActiveX 来创建 XMLHttpRequest 对象。否则，就要使用本地 JavaScript 对象技术来创建。可以通过以下函数创建 XMLHttpRequest 对象：

```
var xmlHttp;
function createXMLHttpRequest()
{    if (window.ActiveXObject)
       xmlHttp = new ActiveXObject("Microsoft.XMLHTTP");
     else if (window.XMLHttpRequest)
       xmlHttp = new XMLHttpRequest();
}
```

在这里，变量 xmlHttp 保存了 XMLHttpRequest 对象新创建的一个实例，如果创建失败，xmlHttp 的值将为 null。通过该对象的属性和方法，可以很容易地与服务器实现请求的发送和响应的接收。

3. XMLHttpRequest 对象的属性和方法

XMLHttpRequest 对象常用属性及方法见表 5-29。

表 5-29　XMLHttpRequest 对象常用属性及方法

属　性	
onreadystatechange	状态改变时发生的事件，可以将它与一个 JavaScript 函数绑定
readyState	请求的状态。有 5 个可能的取值(0:未初始化，1:正在加载，2:已加载，3:交互中，4 :完成)
responseText	服务器的响应，表示为一个串
responseXML	服务器的响应，表示为 XML。可以解析为一个 DOM 对象
status	服务器的 HTTP 状态码。例如，200:OK，404:Not Found，等等
statusText	HTTP 状态码的相应文本。例如，OK 或 Not Found(未找到)等
方　法	
open(method, url, asynch)	建立对服务器的请求，即设置对应服务器端的异步通信程序，URL 为服务器端程序的 URL
send(content)	向服务器发送请求
setRequestHeader (header, value)	把指定首部设置为所提供的值。在设置任何首部之前必须先调用 open()
abort()	停止当前请求
getAllResponseHeaders()	把 HTTP 请求的所有响应首部作为键/值对返回
getResponseHeader(header)	返回指定首部的串值

下面详细介绍这几个方法的应用。

(1) void open(string method, string url[, boolean asynch][, string username][, string password]):建立对服务器的调用,这是初始化一个请求的纯脚本方法。参数说明如下。

① method:必选参数,提供调用的特定方法(GET、POST 或 PUT)。

② url:必选参数,所调用资源的 URL,即服务器端处理程序。

③ asynch:必选参数,指示这个调用是异步的还是同步的。默认值为 true,如果为 false,处理器就会等待,直到从服务器返回响应为止。

④ username:可选参数,表示用户名,用于身份验证。

⑤ password:可选参数,表示用户密码,用于身份验证。

(2) void send([content]):向服务器发出请求。如果请求声明为异步的,这个方法就会立即返回,否则它会等待直到接收到响应为止。可选参数可以是 DOM 对象的实例、输入流或串。传入这个方法的内容会作为请求体的一部分发送。

(3) void setRequestHeader(string header, string value):HTTP 请求中一个给定的首部设置值。它有两个参数:header 表示要设置的首部;value 表示首部的值。需要说明,这个方法在调用 open()之后才能调用。

(4) void abort():停止请求的方法。

(5) string getAllResponseHeaders():返回所有响应的 Http 头,首部包括 Content-Length、Date 和 URI。

(6) string getResponseHeader(string header):返回指定的首部值。

【例 5-16】 设计一个用户注册的页面,使用 AJAX 技术完成用户账户的检测。

分析:用户账户注册是 Web 开发中最常用的功能,在填写表单详细信息以前,检测申请的账户是否存在,可以避免提交表单后,发现账户已经存在而使得用户重新填写。

设注册页面(局部)设计如图 5-16 所示。

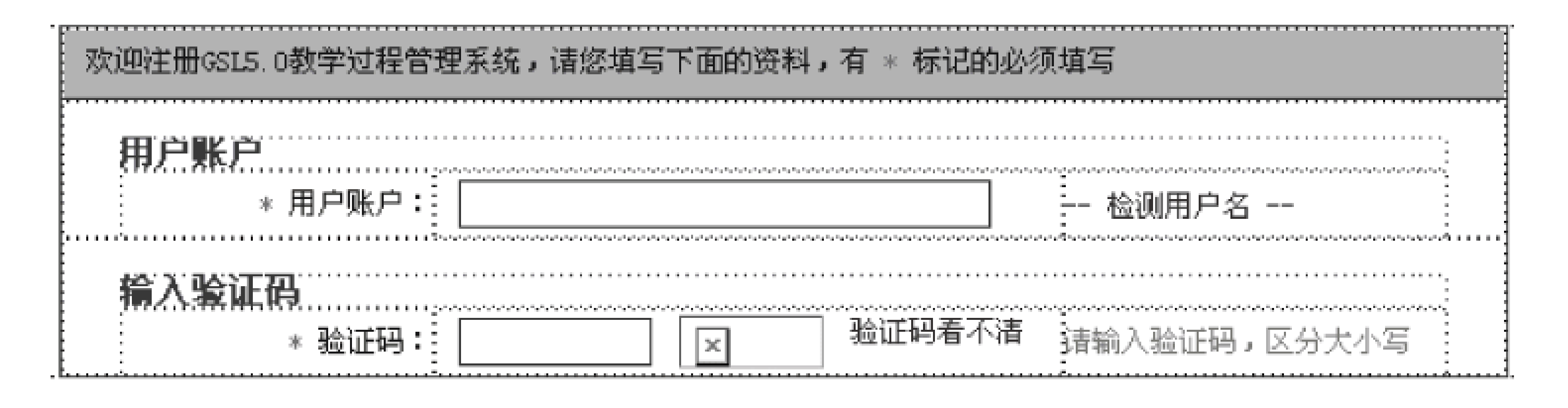


图 5-16 设计中的用户注册页面(部分)

用户注册页面涉及四个文档:①注册页面 newuser-reg.htm;②用户名检测页面 newuser-checkaccount.jsp;③验证码显示 createrandomcode.jsp;④用户信息保存页面 newuser-save.jsp。后三个页面均为服务器页,有关 AJAX 的是注册页面。相关代码如下。

(1) 注册页面 newuser-reg.htm 代码清单:

```
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312" />
<link rel = "stylesheet" type = "text/css" href = "newuser.css" />
```

```
<script type="text/javascript">
//////////////////////////////////////////////////////////////////////////////////
function DispAccountNote()
{
    CheckResult.innerHTML = "用户账户是由字母、数字构成,不能包含汉字字符";
}
function DispAccountCheck()
{
    CheckResult.innerHTML = "<a href='#' onClick='CheckAccount(form1.useraccount)'>-- 检测
用户名 --</a>";
}
//////////////////////////////////////////////////////////////////////////////////
function CheckAccount(obj)
{
   var useraccount = obj.value;
   if(useraccount == "")
   {
      window.alert("用户账户不能为空!");
      obj.focus();
      return false;
   }

    if (escape(form1.useraccount.value).indexOf("%u")! = -1)
    {
      alert("用户账户不能包含汉字,请重新输入");
      obj.focus();
      return false;
    }

    //账户验证,由6~20位的字母、数字、下划线、句点、@,组成,不能包含汉字
    var Expression = /^[A-Za-z0-9]{1}([A-Za-z0-9]|[._@]){1,19}$/;
    var objExp = new RegExp(Expression);
    if (objExp.test(useraccount) == false)
    {
      alert("用户账户由6~20位的字母、数字、下划线、点"."、"@"组成,并且首字符为字母或
数字.");
      obj.focus();
      return false;
     }
    CheckResult.innerHTML = "请等待…";
    //初始化对象并发出XMLHttpRequest请求
    http_request = false;
    if (window.XMLHttpRequest)
    {   // Mozilla或其他除IE以外的浏览器
        http_request = new XMLHttpRequest();
        if (http_request.overrideMimeType)
            http_request.overrideMimeType("text/xml");
    }
    else
     if (window.ActiveXObject)
     {   // IE浏览器
```

```
            try {
              http_request = new ActiveXObject("Msxml2.XMLHTTP");
              }
             catch (e) {
              try {
                http_request = new ActiveXObject("Microsoft.XMLHTTP");
              }
             catch (e) {}
             }
       }
       if (!http_request)
       {
            alert("不能创建 XML HTTP 实例!");
            return false;
        }
        //指定响应方法
        http_request.onreadystatechange = CheckReturn;
        //发出 HTTP 请求
        http_request.open("GET", "newuser-checkaccount.jsp?useraccount=" + useraccount,
true);
        http_request.send(null);
}
////////////////////////////////////////////////////////////////////////////////////
//处理服务器返回的信息,即 jsp 页面的输出
function CheckReturn()
{
    if (http_request.readyState == 4)
    {
      if (http_request.status == 200)
         CheckResult.innerHTML = convert(http_request.responseText);
      else
         alert(http_request.status);
    }
}
////////////////////////////////////////////////////////////////////////////////////
function convert(str)
{
    if (str! = "")
        str = str.replace(/\r/g,"");
    return str;
}
////////////////////////////////////////////////////////////////////////////////////
//重载验证码
function ShowConfirmCode(obj)
{
    var timenow = new Date().getTime();
    obj.src = "createrandomcode.jsp?d = " + timenow;
}
</script>
</head>
```

```
<body style="margin-top:20px">
<form name="form1" action="newuser-save.jsp" method="post" onSubmit="return mysubmit()">
<table class="info-table" cellpadding="0" cellspacing="0">
<tr>
    <td class="row1" height="30">
        欢迎注册 GSL5.0 教学过程管理系统,有<span class="red">*</span>的必须填写
    </td>
</tr>
<!---------------------------- 用户账户 ------------------------------>
<tr>
<td align="center">
<table class="areatable" cellpadding="0" cellspacing="0" border="0">
<tr>
    <td class="areatitle" colspan="3"><strong>用户账户</strong></td>
</tr>
<tr>
   <td width="20%" class="titlecell"><span class="red">*</span>用户账户:</td>
   <td width="40%" class="inputcell">
       <input type="text" name="useraccount" onfocus ="DispAccountNote()"
                  onblur ="DispAccountCheck()" class="text220"/>
   </td>
   <td class="notecell" id="CheckResult">
       <a href='#' onClick='CheckAccount(form1.useraccount)'>-- 检测用户名 --</a>
   </td>
</tr>
</table>
</td>
</tr>
<!---------------------------- 验证码 ------------------------------>
<tr>
<td align="center">
<table class="areatable" cellpadding="0" cellspacing="0" border="0">
<tr>
    <td class="areatitle" colspan="3"><strong>输入验证码</strong></td>
</tr>
<tr>
    <td width="20%" class="titlecell"><span class="red">*</span>验证码:</td>
    <td width="40%" class="inputcell">
      <input type="text" name="confirmcode" value="" align="middle" class="text80"/>
      <img src="createrandomcode.jsp" id="randomcode" width="60px" height="22px">
      <a href="ShowConfirmCode(document.getElementById('randomcode'))">看不清</a>
    </td>
    <td class="notecell">请输入验证码,区分大小写</td>
</tr>
</table>
</td>
</tr>
</table>
</form>
</body>
```

```
</html>
```

(2) 用户名检测页面 newuser-checkaccount.jsp 代码清单:

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.sql.*" %>
<jsp:useBean id="gslpub" scope="page" class="pub.db_gslpub" />

<% //接收客户端提交的数据
String useraccount=request.getParameter("useraccount");
if (useraccount==null) useraccount="";
ResultSet rs = null;
try{
     String strSQL = " select UserAccount from useraccounts where UserAccount = '" + 
useraccount + "'";
     rs=gslpub.executeQuery(strSQL);
     if (rs.next())
       out.println("<font color = '#FF0000'>抱歉!账户[" + useraccount + "]已经被注册
</font>");
     else
       out.println("<font color = '#008800'>祝贺您!账户[" + useraccount + "]可以使用
</font>");
}
catch (Exception ex){
     out.print(ex.getMessage());
}
finally {
     gslpub.disconnectToDB();
}
 %>
```

关于服务端脚本程序,请继续学习下一章的内容。在这里需要说明的就是服务器页的输出将被显示在客户端页面指定的 id 区域。对于其他代码,例如验证码的生成代码、账户保存等服务器代码,在此介绍略,需要的读者可以与作者 E-mail 联系。

5.9 综合举例

Web 客户端的编程比较简单,它不需要特别的编译和运行环境,只要有一个浏览器就可以了。但是,要编写高质量的客户端脚本,并不容易,这取决于大量的实践经验,也来源于用户的需求。没有需求,就不会有深入的编程体验,更无法把一个工具学精。为此,在本章的最后,我们给出了三个在实际 Web 开发中常用的功能,分别用来综合说明 JavaScript 中的窗口控制、图层技术和 HTML DOM 文档的操作,帮助建立总体的 Web 客户端编程思想。

5.9.1 创建折叠式菜单

在许多网页上,都有折叠式菜单,这类菜单的创建可以利用 HTML DOM,通过纯

JavaScript 程序编码实现。目前常用的菜单有折叠式和树形目录结构两种形式。本节介绍折叠式菜单的创建和应用。

设有一个框架页面,分为左右两个 frame,左侧显示一个折叠菜单,右侧是菜单项所对应的页面显示区域,右侧帧名为 mainFrame。

折叠菜单 mainmenu.htm 代码清单:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<style type="text/css">
  a{color:rgb(235,239,71);text-decoration:none; font-size:12px}
  a:hover { color:#FF0000;text-decoration:none}
  .menutitle {
      margin-left:0px; margin-top:7px;margin-bottom:0px;
      background-color:rgb(0,119,166);
      color:rgb(254,254,166);
      width:150px;height:30;
      padding:7px;
      text-align:center;
      font-size:13px;
      border-right:1px solid #204848;
      border-bottom:1px solid #204848;
      border-left:1px solid #e0e0e0;
      border-top:1px solid #e0e0e0;
      cursor:hand;
}
.submenu{
      margin-left:0px;margin-top:0px; margin-bottom:6px;
      background-color:rgb(0,177,234);
      color:rgb(235,239,71);
      width:150px;
      padding:0px;
      text-align:center;
      line-height:200%;
      font-size:13px;
      border-right:1px solid #204848;
      border-bottom:1px solid #204848;
      border-left:1px solid #f0f0f0;
      border-top:1px solid #f0f0f0;
      display:none;
}
</style>
<script>
function SwitchMenu(obj)
{
    var el = document.getElementById(obj);
    var ar = document.getElementById("masterdiv").getElementsByTagName("span");
    if (el.style.display == "block")
        el.style.display = "none";
    else
```

```
        {
            for (var i = 0;i < ar.length;i++)
                if (ar[i].className == "submenu") ar[i].style.display = "none";
            el.style.display  =  "block";
        }
    }
function DispPage(newurl)
{
  eval('parent.frames.mainFrame.location.href = newurl');
}
</script>
</head>
<body>
<div align = "center">
<table width = "170" height = "250" border = "0"cellspacing = "0" cellpadding = "0" bgcolor =
"#FFFFFF">
<tr>
   <td valign = "top" >
   <div id = "masterdiv" align = "left">
   <div class = "menutitle" onclick = "SwitchMenu('sub01')">课程简介</div>
   <span id = "sub01" class = "submenu">
       <a href = "#" onclick = "DispPage('kcjj/kcjj.htm')">教学目标</a><br>
       <a href = "#" onclick = "DispPage('kcjj/jxdg.htm')">教学大纲</a><br>
   </span>

   <div class = "menutitle" onclick = "SwitchMenu('sub02')">教学队伍</div>
   <span id = "sub02" class = "submenu" >
        <a href = "#" onclick = "DispPage('jxdw/hao.htm')">课程负责人</a><br>
        <a href = "#" onclick = "DispPage('jxdw/jxtd.htm')">教学团队</a><br>
        <a href = "#" onclick = "DispPage('jxdw/rkjs.htm')">任课教师</a><br>
   </span>
   </div>
   </td>
</tr>
<tr height = "10">
     <td background = "images/bgpic004.gif"> </td>
</tr>
<tr height = "30">
     <td align = "center" background = "images/bgpic004.gif">
          <select size = "1"onChange = "if (this.value! = '-1') window.open(this.value);">
               <option value = "-1" selected = "selected">    友 情 链 接    </option>
               <optgroup label = "校内站点">
                  <option value = "http://www.sdu.edu.cn/">山大首页</option>
                  <option value = "http://gsl.sdu.edu.cn/">教学平台</option>
               </optgroup>
               <optgroup label = "搜索引擎">
                  <option value = "http://www.baidu.cn/">百度搜索</option>
               </optgroup>
        </select>
     </td>
</tr>
```

```
</table>
</div>
</body>
</html>
```

在浏览器中打开上述网页，显示创建的折叠菜单如图 5-17 所示。

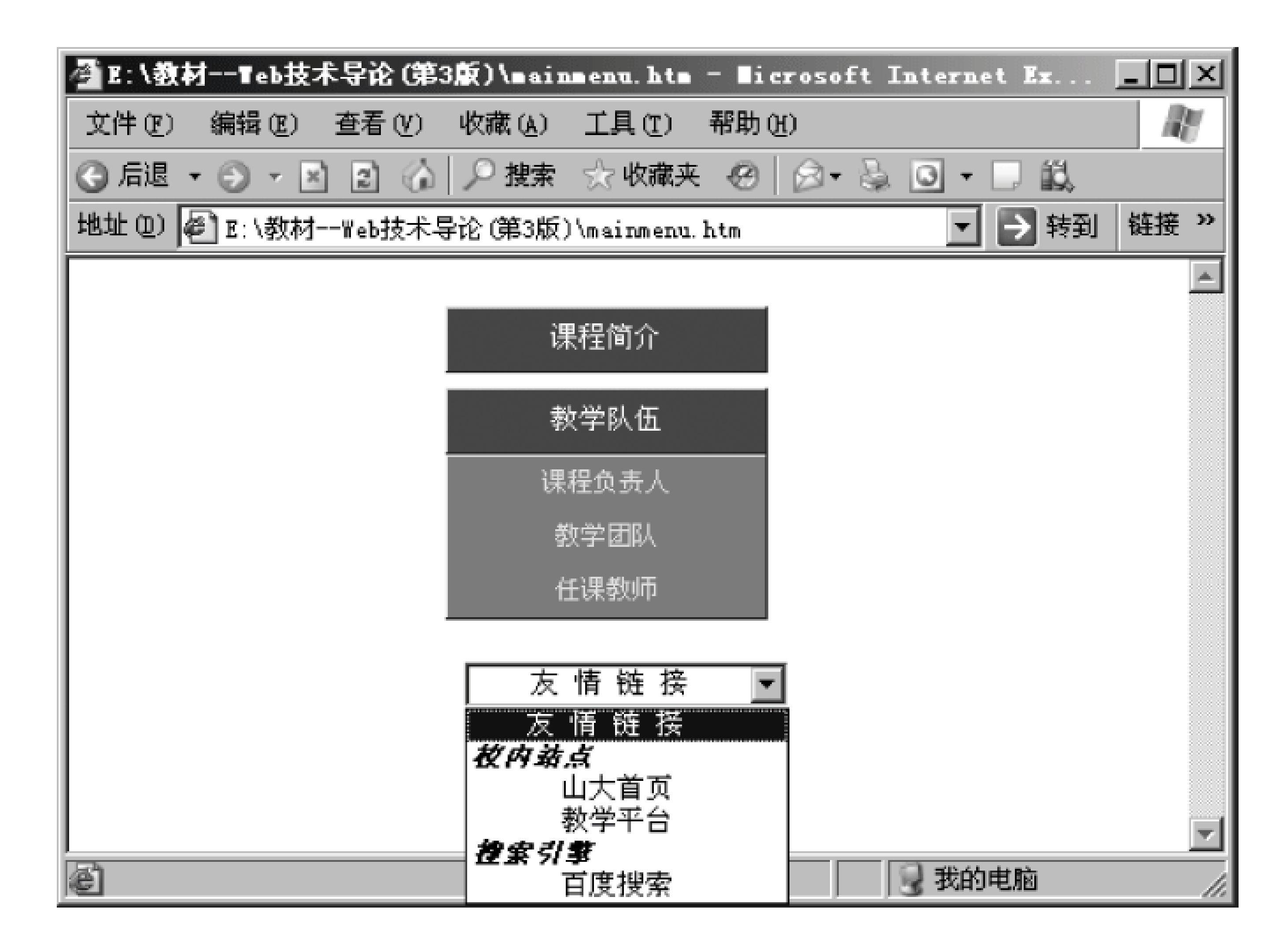


图 5-17 折叠式菜单示例

当用户在一级菜单上单击时，将打开对应的二级菜单。二级菜单对应具体的超链接，当用户将鼠标指针指向二级菜单项目时，在浏览器状态栏中可以看到对应的超链接。根据需要，修改每一个二级菜单项目对应的超链接，就可以很好地实现页面之间的控制。

5.9.2 创建树形菜单

属性菜单也是 Web 开发中常用的菜单，下面通过图层的方式来演示属性菜单的创建。菜单分为二级和三级菜单，要实现的显示界面如图 5-18 所示。

对应上述树形菜单的 menutree.htm 代码清单如下：

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset =
gb2312">
<style type = "text/css">
a {color: #0000FF;text-decoration:none;font-size:14px}
a:hover{color: #FF0000;text-decoration:none}
img{align:absmiddle}
.menutext01{font-size:14px;color: #700000; vertical-align:4px;line
-height:10%}
.menutext02{font-size:14px;color: #005000;vertical-align:4px;line
-height:10%}
.menutext03{ vertical-align:4px;}
</style>
```

新闻公告管理
内容管理
新闻公告列表
新建公告
修改删除
内容审核
系统用户管理
用户组管理
用户账号管理
功能模块管理
功能模块列表
添加修改删除
权限授权审核
论坛管理

图 5-18 树形菜单

```
<script>
////////////////////////////////////////////////////////////////////////////////
function turnit(ss,ii,bb,bott)
{
   bott = bott + 0;
   if (ss.style.display == "none")
   {
       ss.style.display = "";
       ii.src = "images/tmfopen.gif";
       if (bott == 1)
          bb.src = "images/tmminusbottom.gif";
       else bb.src = "images/tmminus.gif";
     }
  else
  {
       ss.style.display = "none";
       ii.src = "images/tmfclosed.gif";
       if (bott == 1)
          bb.src = "images/tmplusbottom.gif";
       else bb.src = "images/tmplus.gif";
  }
}
</script>
</head>
<body style = "font - family:宋体;font - size:13px;overflow - x:hidden;">
<table width = "100 % " cellSpacing = "0" cellPadding = "0" border = "0">
<tr valign = 'middle' height = '16'>
<td onmouseup = 'turnit(Menu03,Img03,tag03,1);'  style = 'cursor:hand;'>
< img id = 'tag03' src = 'images/tmplusbottom.gif'>< img id = 'Img03' src = 'images/tmfclosed.
gif'>
<span class = 'menutext01'>功能模块管理</span>
</td>
</tr>
<tr>
<td id = 'Menu03' style = 'display:none'>
<table width = '100 % ' border = '0' cellspacing = '0' cellpadding = '0'>
<tr height = '16'>
<td>
< img src = 'images/tmnoline.gif'>< img src = 'images/tmjoin.gif'>
<span class = 'menutext03'><a href = 'modules - admin.jsp'  target = 'mainFrame'>功能模块列表
</a></span>
</td>
</tr>
<tr height = '16'>
<td>
< img src = 'images/tmnoline.gif'>< img src = 'images/tmjoin.gif'>
<span class = 'menutext03'><a href = 'modules - admin.jsp'  target = 'mainFrame'>添加修改删除
</a></span>
</td>
</tr>
<tr height = '16'>
```

```
<td>
<img src='images/tmnoline.gif'><img src='images/tmjoin.gif'>
<span class='menutext03'><a href='modules-admin.jsp'  target='mainFrame'>权限授权审核
</a></span>
</td>
</tr>
<tr height='16'>
<td onmouseup='turnit(Menu03_02,Img03_02,tag03_02,1);'style='cursor:hand;'>
<img src='images/tmnoline.gif'><img id='tag03_02'  src='images/tmplusbottom.gif'><img id
='Img03_02' src='images/tmfclosed.gif'>
<span class='menutext02'>论坛管理</span>
</td>
</tr>
<tr>
<td id='Menu03_02' style='display:none'>
<table width='100%' border='0' cellspacing='0' cellpadding='0'>
<tr height='16'>
<td>
<img src='images/tmnoline.gif'><img src='images/tmnoline.gif'><img  src='images/tmjoin.
gif'>
<span class='menutext03'><a href='mybbs/bbsindex.jsp'  target='mainFrame'>论坛维护</a></
span>
</td>
</tr>
<tr height='16'>
<td>
<img src='images/tmnoline.gif'><img src='images/tmnoline.gif'><img  src='images/
tmjoinbottom.gif'>
<span class='menutext03'><a href='mybbs/bbsindex.jsp'  target='mainFrame'>登录论坛</a></
span>
</td>
</tr>
</table>
</td>
</tr>
</table>
</td>
</tr>
</table>
</body>
</html>
```

对于上述树形菜单，为了实现一级、二级、三级菜单的显示和隐藏，使用了多重表格嵌套，读起来非常麻烦，为什么不使用<div>或<tbody>呢？说明如下。

(1) 在表格中，可以对行或单元格进行显示和隐藏控制，如果要对多行进行显示和隐藏，可以使用<tbody>，但是，遗憾的是<tbody>不支持嵌套使用。

(2) 对于<div>和<table>两者的联合使用，不能将多行定义为一个<div>，<div>在表格中的应用，只能出现在<td>中。

以上两点决定了树形菜单 html 代码只能使用<table>的嵌套，如果一行是一级或二

级菜单,它所包含的菜单项就定义在一个单元格中,对单元格进行显示和隐藏。由于结构相似,可以编写菜单创建程序,通过定义菜单数据结构,自动生成上述代码。

5.9.3 数据有效性验证概述

在 Web 开发中,表单作为客户端输入数据的用户界面,为了保证数据输入的有效性,在提交表单以前,往往需要在客户端进行数据的有效性检查,这样可以有效地减少服务器的负载,提高整个 Web 系统的运行效率。

1. 表单的提交

在 HTML 中,表单提交有以下两种方法。

(1) 通过<form>的 onsubmit 事件属性,编写相应的表单输入处理函数。如果函数返回 true,则表单提交;返回 false,则表单不提交。使用该方法,表单中需要包含“提交”按钮,即,type="submit"的按钮。

(2) 通过普通按钮或超链接,通过定义 onclick 事件属性,提交表单。此时,对应的事件函数定义中,可分成两部分,前面是有效性验证,然后通过 Form 对象的 submit()方法提交,即:表单名.submit()。该种方法和第一种方法相比,其程序可读性更好。

在传统的程序中,通常会定义一些快捷键,也可以对表单提交定义快捷键。方法如下。

(1) 定义键盘操作处理函数:

```
function doByKey()
{
    if ( window.event.keyCode == 113) //F2,提交表单
      form1submit();
    if (window.event.keyCode == 27)    //Esc,放弃
      form1cancel();
}
```

(2) 在<body>标记中,添加 onkeydown 事件属性。代码如下:

```
<body onkeydown = "doByKey()">
```

2. 数据的有效性验证

在表单提交前,通常需要验证数据的有效性。验证数据有效性除了采用传统的程序编码外,采用正则表达式验证数据有效性的效率更高,编程量较少。采用正则表达式验证数据的有效性,有三个步骤:①书写正则表达式;②创建正则表达式对象;③验证数据。

例如,要验证一个 E-mail 是否正确,可以编写下面的代码:

```
function checkemail(str)
{
    //在 JavaScript 中,正则表达式只能使用“/”开头和结束,不能使用双引号
var Expression = /\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*/;
var objExp = new RegExp(Expression);
if (objExp.test(str) == true)
    return true;
```

```
else
    return false;
}
```

在完成一种类型数据的有效性验证的函数后，可以修改表单提交函数，使用各个表单域的有效性验证函数，即：

```
<input type="button" value="确定" onclick="form1submit()">
```

表单提交函数事件 onclick 对应的函数 form1submit ()代码形式如下：

```
function form1submit ()
{
    if  (myform.email.value == "")
    {
        alert("请输入 E-mail 地址!");
        myform.email.focus();
        return false;
    }
    if (!checkemail(myform.email.value))
    {
        alert("您输入的 E-mail 地址不正确!");
        myform.email.focus();
        return false;
    }
    //其他输入域验证
    form1.submit();
}
```

当用户单击"确认"按钮时，执行用户定义的表单检查程序，检查各个输入域数据的有效性，如果数据全部有效，最后执行默认的表单处理函数，即提交表单。

5.9.4 页面安全性设置

在 Web 开发中，经常会对打开的网页做一些安全性的设置，例如禁止用户复制网页内容、禁止网页另存为、禁止用户刷新屏幕、屏蔽有关键盘事件、防止 SQL 注入攻击、使用验证码登录等。下面是一些常见的页面安全性设置。

1. 禁止用户复制网页内容

有时候可能不希望用户复制网页内容，或将网页另存为，其实现方法非常简单，主要是添加<body>标记的有关事件属性即可。相关的事件属性有：

(1) onselectstart，对象将要被选中时触发。如果设置<body onselectstart="return false">，则用鼠标左键选择页面内容的操作将被禁止。

(2) oncontextmenu，单击时触发。

可以用于<body>标记和许多其他的标记，如果要禁止在某个元素上右击，可以赋值 false，例如：<body oncontextmenu="alert ('快捷菜单被禁用');self. event. returnValue=false">，则用户在网页上右击时弹出一个警告对话框，而不是原先的快捷菜单。

综合上面的事件,则下面的<body>标记属性将禁止几乎所有的页面操作:

```
<body onmouseup = document.selection.empty() oncontextmenu = "return false"
onselectstart = "return false" ondragstart = "return false" onbeforecopy = "return false"
oncopy = document.selection.empty()>
```

如果希望禁止网页另存为操作,可以在文档体内,添加下面的标记:

```
<noscript><iframe src = ""></iframe></noscript>
```

当用户在浏览器中选择“另存为”命令时,显示“无法保存该网页”的错误提示对话框。此时可以通过“查看”→“查看源文件”命令查看页面内容。如果不希望用户查看页面源文件,可以进一步设置。

上述事件属性,不仅可以作用于<body>标记,同样也可用于其他标记中,例如,可以设置<td>标记的 oncontextmenu 事件属性。

2. 禁止用户屏幕刷新

在 Web 开发中,当用户填写表单时,如果网速较慢,用户往往会习惯性地单击“刷新”按钮,这将导致用户重复提交订单,从而产生错误。此时,应该将浏览器的菜单屏蔽,这可以通过 JavaScript 来实现。下面通过一个例子来讲解其实现方法。

在页面 goods.htm 中,有一个超链接“填写订单”,代码清单(goods.htm)如下:

```
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
<title>屏蔽浏览器主菜单</title>
<script language = "javascript">
function deal(myurl)
{
  window.opener = null;
  window.close();
  window.open(myurl,'','width = 580,height = 510');
}
</script>
</head>
<body>
  <a href = "#" onClick = "deal('order.htm');">填写订单</a></td>
</body>
</html>
```

页面 order.htm 为要打开的订单页面。当单击“填写订单”超链接时,将打开该网页。

3. 屏蔽有关键盘事件

在表单提交时,除了不能够进行刷新页面操作外,通常情况下,也不允许用户通过键盘刷新屏幕(F5)、执行后退等操作。要实现上述功能,需要在订单页面(order.htm)中,通过屏蔽键盘 F5 键、Enter 键、退格键、Ctrl+N 快捷键等来完成。

屏蔽键盘的相关事件代码(order.htm)如下:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<script type="text/javascript">
function mykeydown()
{
  if(event.keyCode == 8){
    event.keyCode = 0;
    event.returnValue = false;
    alert("当前设置不允许使用退格键");
  }
  if(event.keyCode == 13){
    event.keyCode = 0;
    event.returnValue = false;
    alert("当前设置不允许使用 Enter 键");
  }
  if(event.keyCode == 116){
    event.keyCode = 0;
    event.returnValue = false;
    alert("当前设置不允许使用 F5 刷新键");
  }
  if ((event.altKey)&&((window.event.keyCode == 37)||(window.event.keyCode == 39))){
    event.returnValue = false;
    alert("当前设置不允许使用 Alt + 方向键←或方向键→");
  }
  if ((event.ctrlKey)&&(event.keyCode == 78)){
    event.returnValue = false;
    alert("当前设置不允许使用 Ctrl + N 新建浏览器窗口");
  }
  if ((event.shiftKey)&&(event.keyCode == 121)){
     event.returnValue = false;
     alert("当前设置不允许使用 Shift + F10");
  }
}

function click()
{
   event.returnValue = false;
   alert("当前设置不允许使用右键!");
}
document.oncontextmenu = click;
</script>
</head>
<body onkeydown = "mykeydown()">
屏蔽部分键
</body>
</html>
```

最后需要说明的是，用户在浏览网页时，如果在有些网页中，不能使用鼠标拖动的方法选中文字，或不能右击，通常是因为网页中设置了<body>的相关属性，此时，可以通过设

置浏览器关于JavaScript的相关设置来解决。具体方法是：选择“工具”→“Internet选项”命令；在打开的“Internet选项”对话框中，选择“安全”选项卡，单击“自定义级别”按钮，打开“安全设置”对话框；在“安全设置”对话框中，将“活动脚本”和java设置为“禁用”，来禁止JavaScript的运行。然后按F5键刷新网页。此时，网页中的内容就可以复制了。最后，为了保证浏览器正常工作，复制完成后，应恢复浏览器的默认设置。

本章小结

本章首先介绍了程序设计的基本概念，目的是使读者从思想上对编程有一个基本的认识。然后讲解了浏览器的基本工作原理，概要介绍了JavaScript程序设计语言的一般语言要素，包括基本字符、数据与数据类型、表达式、语句和函数等，这是所有程序设计语言都共有的成分，也是本章JavaScript和下一章JSP服务器编程的语言基础。只有程序设计语言的语句是不能编写复杂的程序的，还需要一个开发环境，也就是说需要特定的函数库或类库。因此，接下来讲解了JavaScript编程用到的标准库，这就是JavaScript内部对象及函数、浏览器对象和HTML文档对象。最后，讲解了Web交互中表单输入的数据获取、数据的有效性验证、页面之间的交互等大量的实例，以及多个综合例子，来讲解这些内部对象的功能和使用，所讲述的代码都来源于我们的研发项目，是Web开发中最具共性的内容，有很好的实用性。

习题5

一、简答题

1. 如何理解浏览器和客户端脚本引擎之间的关系？

2. JavaScript语言有哪几个组成部分？简述各个部分的功能。

3. 在JavaScript中，myArray = new Array(10)是什么意思？如何定义一个3×4的二维数组？

4. 什么是网页对象？简述window对象和document对象常用的属性和方法。

5. 画出HTML DOM的对象层次图。文档对象document有哪些常用的属性和方法？

6. 什么是IntelliSense技术？如何知道一个html标记或一个JavaScript内部对象有哪些属性或方法？

7. 有哪些常用的浏览器事件？它们是如何触发的？

8. 如何设置body对象，以禁用页面上的右键来防止复制页面内容？

9. 如何设置一个“关闭窗口”超链接，使得单击该超链接时，不弹出询问对话框“要关闭窗口吗？”

10. 什么是正则表达式？写一个正则表达式，实现验证密码是否由6～20位的字母、数字、下划线和英文句点(.)组成的正则表达式。

11. 对于表单输入，编写脚本程序，完成下列功能：

(1) 不提交表单，检测输入密码是否一致。

(2) 通过单击一个复选框，来实现一组复选框的全选或取消操作。

(3) 编写检查在 E-mail 输入框中输入的 E-mail 格式是否正确的函数。

(4) 编写检查在电话号码输入框中输入的电话号码是否正确的函数。

12. 在表单输入中，默认情况下，按 Enter 键，则提交表单，如何设置 onKeyPresss 事件属性，使得按 Enter 键后，使输入焦点移向下一个表单元素，而不是提交表单？

13. 如何在两个 htm 页面之间传递参数？举例说明。

14. 什么是网页对话框？使用网页对话框，打开一个便条输入页面，将输入内容在父窗口中列表显示。

15. 在 Web 开发中，对于网页的安全性，回答下列问题：

(1) 要禁止用户复制网页内容，如何设置？

(2) 要禁止用户执行“另存为”命令，如何设置？

(3) 要禁止屏幕刷新，如何设置？

(4) 要屏蔽 F5 键，如何设置？

(5) 在浏览网页时，如果页面设置了禁止用户“另存为”，如何复制网页内容？

二、阅读理解题

1. 阅读下面的代码，写出运行结果。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>
<body>
<select name="sel"onchange="document.write(document.all.sel.value);"></select>
<script type="text/javascript">
var counts=0;
var arr = new Array("text1","text2","text3","text4");
counts=arr.length;
for (i=0;i<counts; i++)
{
    document.all.sel.options[i] = new Option(arr[i],("val"+i));
}
</script>
</body>
</html>
```

2. 说明下列函数的功能。

```
//参数 str 为一个字符串
function convert(str)
{
   if (str! = "")
   {
     str=str.replace(/\r/g,"<br>");
     str=str.replace(/\s/g," ");
   }
   return str;
}
```

三、编程题

1. 编写一个 JavaScript 函数,比较两个字符串是否相等,如果相等,返回 true,否则返回 false。

2. 利用 div 图层技术,设计一个总是置于页面右上角的图片广告,即当用户单击垂直滚动条时,图片一直在浏览器窗口的右上角。

3. 编写一个程序,在客户区的中央显示一个 300×200 大小的图层,并且可以用鼠标拖动图层。

4. 编写一个将网页中的表格导出为 Excel 表并打印的程序。

5. 编写一个利用 CSS 样式实现打印页面指定内容和分页打印功能的程序。

第6章 服务端编程

【本章导读】

对于Web应用，大量的数据管理工作和业务逻辑都是通过服务端程序实现的。服务器程序是在Web服务器上运行的，不同的Web服务器，其编程语言和开发工具也不相同。目前，常用的Web服务器编程语言有ASP、JSP和PHP，它们分别适用于不同的Web服务器。例如Windows平台中的Internet信息服务(IIS)，支持ASP开发。如果要使用JSP开发Web应用，则需要在Web服务器上安装Tomcat应用服务器。在Web应用的开发中，Java技术以其平台无关性受到了开发人员的欢迎，作为Java技术的一种实现，结合Servlet和JavaBean，使得JSP成为众多Web应用首选的开发工具。

本章围绕Java技术，以JSP编程为例，详细介绍Web应用中服务器端的编程问题。主要内容包括Java程序设计基础以及JSP技术两个方面。在Java程序设计中，概要性地介绍了Java程序设计中的概念，包括语言基础及常用的Java包，为JSP编程做好概念上的铺垫。在JSP技术中，以任务驱动的方式，讲解了服务器开发中遇到的共性问题及解决办法，包括JSP中的数据类型及其转换、数组与集合类、文件操作、JSP内置对象、JSP中的参数传递方法以及JDBC与数据库编程。这些内容是每一个Web应用中都可能遇到的问题，在讲解中给出了许多实用的代码和JavaBean类，最后给出了三个综合案例，并进行了讲解。

【知识要点】

6.1节：B/S三层体系结构、服务端脚本引擎、服务端脚本程序。

6.2节：Java程序设计语言的特点、类的概念、类的定义、封装、抽象、对象、构造函数、包、接口、Java基础类库、Java Applet、Java Servlet。

6.3节：JavaBean的概念。

6.4节：JSP的运行环境、JSP指令、JSP声明、JSP中的数据类型及其转换、数组、JSP内置对象、在JSP中使用JavaBean。

6.5节：JDBC接口、JDBC数据库驱动程序、JDBC API、java.sql包、SQL、数据库操作。

6.6节：文件上传、表单数据处理、页面间的数据传递。

6.7节：Web系统分析、功能设计、数据库设计、界面设计、使用AJAX。

6.8节：Web系统开发环境的概念、JDK开发环境简介、NetBeans开发环境简介、Eclipse开发环境简介。

6.1 B/S 三层体系结构与 Web 服务器脚本程序

服务端脚本程序是指在 Web 服务器上运行的程序。在 B/S 模式下，当用户下载一个网页时，如果网页中包含服务端脚本程序，Web 服务器将调用服务端脚本程序引擎(如 Tomcat 应用服务器)执行网页中的脚本程序，然后把执行的结果网页发送到客户端浏览器显示。

6.1.1 B/S 三层体系结构

在 Internet 环境，客户机/服务器(C/S)两层体系结构表现出了许多局限性，它需要在企业局域网内运行，缺少可伸缩性，客户端维护复杂等。为此，出现了基于 Web 的浏览器/服务器(B/S)三层体系结构(3-tier)。这种体系结构大致可以定义为客户机上的表示层、中间的 Web 服务器层和后端的数据库服务器层。B/S 三层体系结构如图 6-1 所示。

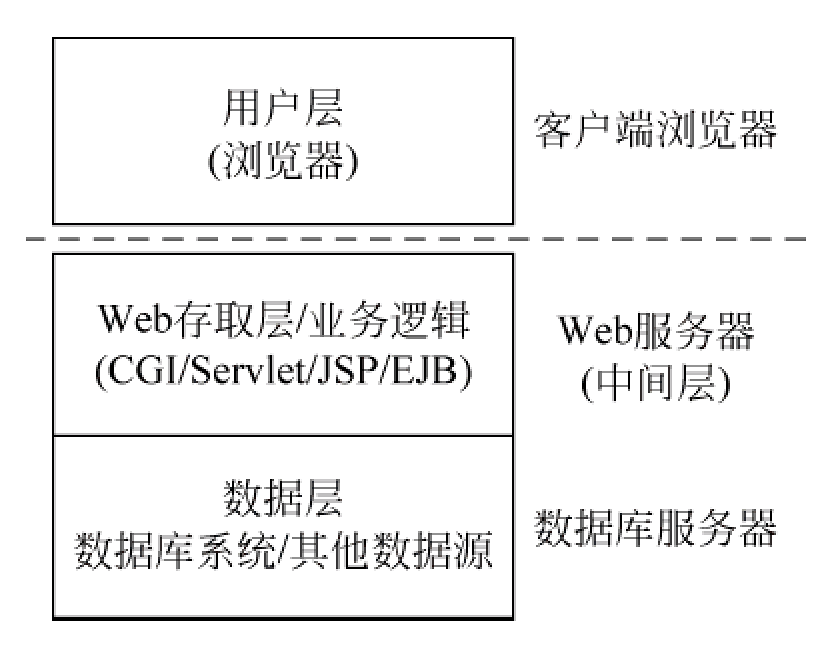


图 6-1 B/S 三层结构的分层功能界定

在 B/S 三层体系结构模式下，客户端不再需要安装特定的客户端应用程序，取而代之的是通用的 Web 浏览器软件，所有的用户业务逻辑都被部署在新的中间层上。新的中间层往往是一组公共网关接口(Common Gateway Interface，CGI)程序，CGI 程序充当一种中间软件，从 Web 浏览器接收请求，决定必须调用哪些计算资源来满足这些请求，并向浏览器发回响应。传统的 CGI 程序一般是由 C/C++开发的，随着 Java Servlet 的出现，CGI 的功能更多地是由 Java Servlet 来实现，它具有更高的效率和可移植性。

由于三层体系结构通常是基于 Web 的，因此中间层应用程序通常工作在 Web 服务器上，被视为 Web 服务器的一种功能扩展，因此中间层又称为 Web 服务器层。在 Web 服务器上，通过大量的包含 CGI/Servlet 的服务器脚本程序页面，接收来自客户端浏览器的请求，以及完成对数据库的操作。

6.1.2 脚本引擎与服务端脚本程序

什么是脚本引擎呢？简单地讲，脚本引擎就是指脚本程序的运行环境，负责脚本程序的解释，来具体处理用相应脚本语言书写的脚本命令。例如 ASP 脚本语言必须运行在 IIS 上；Tomcat 是 JSP 和 Servlet 的容器，运行 JSP 网页必须安装和配置 Tomcat。没有脚本引擎，脚本程序将不能运行。

在 ASP 结构中，ASP 解释器(ASP. DLL)负责 ASP 页内服务器端脚本程序的解析任务。这需要安装相应脚本程序语言的脚本引擎，即脚本程序解释器，来具体处理用相应语言书写的脚本命令，它们以 COM 组件的形式供 ASP 解释器调用。Active Server Pages 带有两个脚本引擎——Microsoft Visual Basic Scripting Edition (VBScript) 和 Microsoft

JScript，其中 ASP 中的默认语言是 VBScript。在 Windows 平台上，Internet 信息服务器 IIS 内含 Active Server Pages，安装 IIS 后，VBScript 和 JScript 脚本引擎会自动地安装在服务器上。

如果要使用其他的脚本程序语言，例如 JSP，则必须在 Web 服务器上安装相应的脚本引擎，即 Tomcat 应用服务器。脚本引擎遵循 ActiveX 脚本标准并作为一个 COM（组件对象模型）对象驻留在 Web 服务器上。

在 Web 服务器上安装服务端脚本引擎后，进行简单的配置，然后就可以在网页中编写服务端脚本程序了。与客户端脚本程序书写在<script></script>标记对内不同，服务端脚本程序一般书写在定界符“＜％”和“％＞”内。

【例 6-1】 服务器页面与 JSP 程序的运行示例。

分析：服务端脚本程序通常被书写在服务器页中，例如 JSP（Java Server Page）、ASP（Active Server Page）等。当用户要浏览服务器页时，Web 服务器将调用相应的应用服务器执行服务器页中包含的服务器脚本程序，最后将执行结果返回给用户浏览器。

一个包含服务端脚本程序的网页（test.jsp），代码清单如下：

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%!
String datestr = "";
 %>
<html>
<body>
<%
java.util.Date nowdate  =  new java.util.Date();
java.text.DateFormat df  =  new java.text.SimpleDateFormat("yyyy - MM - dd HH:mm");
datestr  =  df.format(nowdate);
datestr  =  datestr.substring(0,16);
 %>
现在的时间是：<%  = datestr %>
</body>
</html>
```

这是一个典型的 JSP 文件，可以看出，它是 Java 程序和 HTML 标记的混合体。看起来比较乱，需要一个习惯的过程。此外，由于程序代码和 HTML 标记混在一起，需要注意格式，特别是空格，否则也会产生运行错误。例如：＜％＝datestr％＞，“＜％”和“＝”之间不能有空格，有的用户习惯加一个空格，使格式看起来美观，但却会产生错误。通过浏览器下载该页面时，Web 服务器将执行其中的服务端脚本程序，在客户端浏览器中显示执行后的结果。

要运行 JSP 页面，必须要安装 Tomcat 并进行相应的配置，详细内容请见第 2 章。为本章示例程序说明的方便，在 Tomcat 主配置文件\conf\server.xml 中创建虚拟目录，即：＜Context path＝"/book" docBase＝"d:\haobook" /＞，本章所有的要调试的 JSP 页面，如果没有特别说明，均保存在 d:\haobook 文件夹中。要运行上述页面，在浏览器地址栏中输入 http://127.0.0.1/book/test.jsp，运行结果如图 6-2 所示。

选择浏览器的“查看”→“源文件”命令，可以看到服务器页传送到客户端的 HTML 代

码,本例中 JSP 页面生成的 html 代码如下:

```
<html>
<body>
现在的时间是: 2012 - 01 - 28 16:30
</body>
</html>
```

如果 JSP 页面中还包含客户端脚本程序,用户可以通过客户端脚本程序,对服务器上生成的数据进行操作,来完成功能强大的 Web 页面编程。

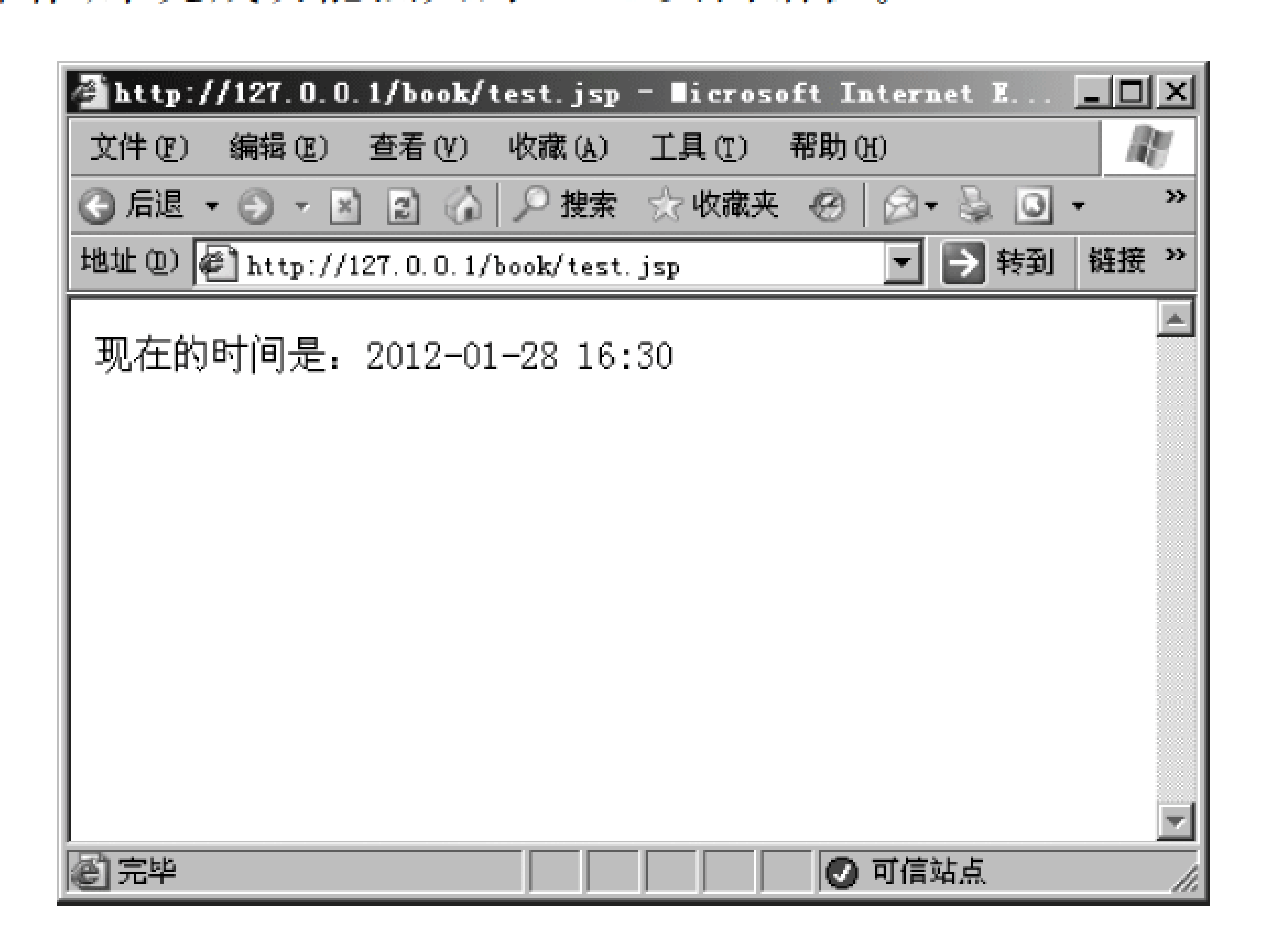


图 6-2 JSP 页面运行结果示例

6.2 Java 程序设计基础

Java 程序设计语言是 Java 技术的重要组成部分,Java 技术是 Sun 公司于 1995 年推出的一种极富创造力的计算平台。Java 技术的多功能性、有效性、平台无关性以及安全性使它成为网络计算领域最完美的技术,给未来的计算模式产生了革命性的影响,是继 HTML 后,Internet 发展的又一个里程碑。今天,Java 技术已经无处不在,从桌面 PC 到科学超级计算机和互联网,从移动电话到移动手持设备,从家庭游戏机到信用卡,几乎在所有的网络和设备上都会看到 Java 技术的身影。

6.2.1 Java 程序设计语言

Java 程序设计语言是 Sun 公司于 20 世纪 90 年代初开始研发的,总设计师为 James Gosling。Java 程序设计语言和 C/C++ 等一般的程序设计语言类似,都包含基本符号、数据、数据类型、表达式、流程控制、类与对象等程序设计的概念。

在 Java 中,基本的语言元素,包括基本符号、数据、数据类型、表达式、流程控制,与 C/C++ 等程序设计语言类似,也和 JavaScript 的语法类似。下面是 Java 语言不同于 C++ 的一些特有性质。

(1) Java 没有主函数和全局函数。C++并非完全意义上的面向对象语言，最明显的例子是，在 C++中，必须有一个独立的主函数(在 DOS 和 UNIX 下是 main()，在 Windows 下是 WinMain())，还可以定义可以直接使用的大量的全局函数或者使用 C++中的命名空间"extern C"来使用原有的 C 过程调用。

Java 是完全面向对象的语言，它没有主函数等完全孤立的东西，任何函数都必须隶属于一个类。当然，任何程序都有一个入口，Java 程序也有主入口函数，名称同样是 main()，但它必须包含在一个类中。一般形式是：

```
public class AppName{
    public static void main(String args[ ])
    {
    … // 代码
    }
}
```

(2) Java 没有全局变量。在 Java 程序中，不能在类外面定义全局变量，只能通过在一个类中定义公用静态变量来实现全局变量。例如：

```
class GlobalVar {
    public static GlobalVarName;    // 全局变量定义
}
```

因为 public static 成员是一种类属成员变量，只要定义了类，其中的类属成员变量就分配空间，而不需要必须声明类的对象，使得其他类可以访问和操作该变量。

可见，在 Java 中，对全局变量进行了更好的封装。而在 C++中，不依赖于任何类的不加封装的全局变量往往会导致系统崩溃。

(3) Java 没有结构和联合。在 C++中，为了保持和 C 的兼容，继续支持结构(struct)和联合(union)。但是，在 Java 中，则完全摈弃了这些面向过程时代的概念。

(4) 字符串不再是字符数组。在 C 和 C++中，字符串操作往往会导致许多内存问题，如内存非法操作、内存泄漏等。因为字符串 char *s 和不定界的字符数组 char s[]是等价的。但两者只是为变量 s 分配一个指向字符串的指针，存储字符串内容的内存需要申请和释放。

在 Java 中，字符串和字符数组已经被分开了。字符串是一个完全意义上的对象，需要用 String 类来定义。

(5) Java 用包(Package)来分解命名空间。在大型的软件工程中，怎样保证程序员之间命名的类不重名，以及如何避免供应商提供的类和程序员自己命名的类不重名呢？虽然有许多方法可以避免重名，但是如果在发现问题以前，工程已经启动了，要修改这些重名问题，就变得非常麻烦。

在 Java 中，引入 Package 概念来解决上述问题。Package 有效地通过集合类来划分命名空间，在不同包内的类可以同名，但不会引起混乱。

Java 并没有彻底解决命名冲突的问题，扩展基类可能引起派生类的冲突。例如：用户派生一个类，增加一个方法 foo。如果以后供应商提供新的版本，在新类中也包含了 foo 方法，冲突就出现了。

(6) Java 没有独立的头文件。在 C++中，每一个.cpp 实现文件都对应一个.h 头文件，

在头文件中,往往包含了.cpp文件中用到的类的定义。在Java中,关于类的一切东西(属性和方法)都被放到一个单独的文件中,类中方法的实现必须在定义的过程中同时进行。

因为类方法的实现代码必须在方法定义时完成,但是一个函数往往是几十行或者几百行的程序代码,这样就使得阅读类很难一下子就看到一个类的全貌。Java的设计者已经考虑到了这个问题,为此,在JDK中提供了两个工具来补偿:Javap来打印类标识,Javadoc为源代码提供标准的HTML文档。

(7) 数据类型。在C/C++中,对于不同的平台,编译器对简单数据类型int、float等分配不同长度的内存空间,例如int在IBM PC中为16b,在VAX-11中为32b,这导致了代码的不可移植性。但是在Java中,对于这些基本数据类型,总是分配固定长度的空间,int总是32b,这就可以保证Java的平台无关性。

在C/C++中通过指针可以进行强制类型转换,这往往会带来不安全性。在Java中,有严格的类型相容性检查。另外,在Java中,没有模板类,而C++中的模板类(参数化类,即形式参数对应的实际参数是数据类型)可以有效地简化程序代码的编写,但不能减少可执行代码的长度。这就意味着,在Java中,只能靠相似代码的手工复制和修改。

(8) 常量修饰符const的使用限制。在C++中,const常量修饰符有着重要的应用,它对于提高代码质量起到了积极的作用。例如,用户可以声明函数参数或者函数的返回值为const类型,这样可以有效地防止在函数内部对函数参数的不正当修改或者对返回值的修改。另外,可以将类的一个成员函数声明为const,表明该方法不能修改它操作的任何对象。

在Java中,支持常量操作符、只读变量,通过final关键字实现。但是,Java没有提供一种机制,使得一个变量在参数传递或者返回的过程中只读,或者定义一个不能修改操作对象的常量方法。上述的省略,为Java程序带来了一个可能引起不正当修改错误的隐患。

另外,在Java中,宏也不再被支持。

6.2.2 类与对象

在20世纪90年代以前,自顶向下逐步求精的结构化程序设计是软件开发的主要方法,直到现在,这种结构化的程序设计思想仍然被广泛地采用。Pascal、C、BASIC、FORTRAN等高级语言很好地实现了结构化编程的思想,通过过程和函数(又称子程序),把一个复杂的问题分解成若干个相对简单的子问题,如果子问题还比较复杂,再继续划分,最后将划分后的每个问题用过程和函数来实现。

20世纪90年代以后,面向对象技术使人们近半个世纪来的软件开发思想产生深刻的变革。这一技术强调利用软件对象进行软件开发,它将自然界中的物理对象和软件对象相对应,建立了类和对象的概念。由于客观世界的实体和软件结构的对象一一对应,从而增加了软件系统的可扩展性和可维护性。面向对象技术将自然界中的物理对象和软件对象对应起来,在传统的数据结构基础上加入了成员函数(方法)的概念,从而赋予对象以动作。

1. 类与对象的概念

类(class)是包含数据和处理这些数据的过程的数据结构。可以将类看成和int、float等基本数据类型一样的数据类型,用它来创建数据对象,它指定了相应内存区域的处理和解释规则。

对象(object)是用类来声明的数据结构,如果将类比作数据类型,对象就是相应数据类型的变量。对象是类的实例,占据确定的内存空间。

2. 类的定义

在 Java 中,用户可以定义一个基类,也可以从别的类进行派生(extends),或者通过实现(implements)一个或多个接口来定义一个新的类。类定义的一般形式是:

```
[类型修饰符] class <类名> [extends <父类名>] [implements <接口名列表>]
{
    成员变量声明;
    成员函数定义;
}
```

其中,类型修饰符决定类的类型,有四种。

(1) abstract,抽象类,必须包含至少一个抽象成员函数。抽象类不能够创建对象,需要用其派生类创建。例如,可以定义一个图形类 CFigure,将其定义为抽象类,然后从其派生具体的图形类,例如三角形类 CTringle、矩形类 CRectangle 等。

(2) public,公有类,能被其他类访问的公有类,在其他包(package)中要使用该类,需要先用 import 导入,否则只能在它定义的 package 中使用。

(3) final,最终类,说明一个类不能再派生子类。

(4) synchronicable,表示所有类的成员函数都是同步的。

上述修饰符可以同时出现两个或以上,但修饰符 abstract 和 final 不能同时使用。

class 为关键字,表明接下来是类的定义,类名是一个用户自定义的标识符。

如果是派生类,需要通过 extends 给出父类,如果实现接口(interface),需要通过 implents 给出接口名,接口可以是一个或多个。

花括号内说明类的成员变量和成员函数,又称类的属性(attribute)和方法(method)。

(1) 声明成员变量。成员变量定义的一般形式是:

```
[访问级别修饰符] <类型> <成员变量列表>;
```

其中,访问级别修饰符为类成员声明访问级别,可以是 private、public、protected。还可以添加的修饰符有 final(最终,初始化后不能再改变其值的变量)、volatile(多线程变量)。这些访问级别可以按任何顺序出现,也可以多次出现。在两个访问级别声明符号之间的成员具有相同的访问控制级别。

如果成员变量包含修饰符 static,此变量称为类变量,不加 static 的变量称为对象变量。如果变量说明时缺省修饰符,则它可被同一包中的所有类访问。

(2) 定义成员函数。成员函数又称类的方法(method),是类中定义的可对类进行操作的函数模块。成员函数定义的一般形式为:

```
[访问级别修饰符] 返回值类型 <方法名>( [形式参数列表] ) [throws 异常列表]
{
// 函数体(Java 程序代码)
}
```

其中修饰符可以是 private、public、protected 和 final(最终,不能由子类改变的方法)、abstract(抽象方法,无方法体)、synchronized(线程同步的方法)、native(本机方法)。修饰符可以有一个或多个。另外,还可以有 static 来声明静态成员函数,表明此方法为类方法,无 static 修饰的方法为对象方法。

"形式参数列表"给出函数的形式参数及类型,形参之间用逗号分隔。当所定义的方法没有形参时圆括号内为空。

3. 封装和抽象

在面向对象技术中,一个主要的目标就是对象的封装和抽象。封装(Encapsulation)是指对象可以拥有私有元素,将内部细节隐藏起来的能力。封装将对象封闭起来,管理着对象的内部状态。而抽象则和对象的外部状态紧密相关,它通常用来描述对象所表示的具体概念、对象所完成的任务以及处理对象的外部接口。抽象处理的是对象的可见外部特征。

在 C 中,通过关键词 static 可以实现有限的封装。当一个变量在一个函数内部被说明成 static 形式时,该变量就只在函数中存在,并且只在函数内部有效。另外,一个全程变量被说明成 static 形式时,该变量只在其所在的文件有效,这样可以避免不同的文件中全局变量的重名。

在 C++ 和 Java 等面向对象的程序设计语言中,类的每一个成员都被说成 public、private 和 protected 型,用这些关键词来实现数据的抽象和封装。

(1) 关键词 public。类中所有 public 成员构成类的接口,它们是类的抽象性的表现。出于简单、经济和安全的愿望,理论上讲,类的公开元素越少越好。但是,类又必须和外部打交道,public 成员是不可缺少的。

(2) 关键词 private。类中的 private 成员只能被类的成员函数、友元类或外部友元函数访问,从而实现类的封装性。在默认状态下,所有的成员都是私有的。

(3) 关键词 protected。在面向对象技术中,派生是类的重要性质。类的 private 成员将不能被派生类中的成员访问,这就大大限制了类的灵活性。类的 protected 成员可以被类的派生类成员访问。

类成员访问规则如图 6-3 所示。

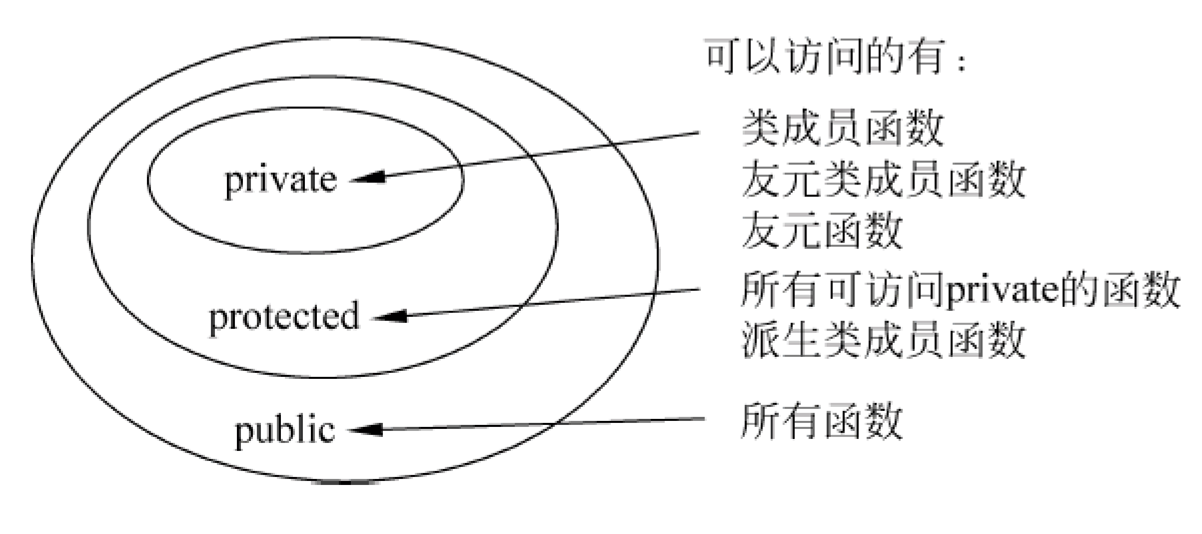


图 6-3 类成员访问规则

4. 静态成员

支持封装和抽象的另外机制还包括关键词 static。当一个成员被说明成 static 时,则该成员在程序中只有一个副本存在,而不是在每个对象中都有一个副本。所有的对象共享类

中的静态成员。在一些面向对象的程序设计语言中，静态成员被称为类变量，或类属变量。

例如，定义一个含有静态成员变量的类如下：

```
class CS {
    static int a;
    int b,c,d;
};
CS objs[3];
```

在类 CS 中，包含四个成员变量，其中成员变量 a 为静态成员变量。数组 objs 包含三个 CS 类对象，由于 CS 类包含一个静态成员变量 a，因此三个 CS 对象共享一个静态成员变量 a，每个对象都包含自己的普通成员变量 b，c，d，如图 6-4 所示。

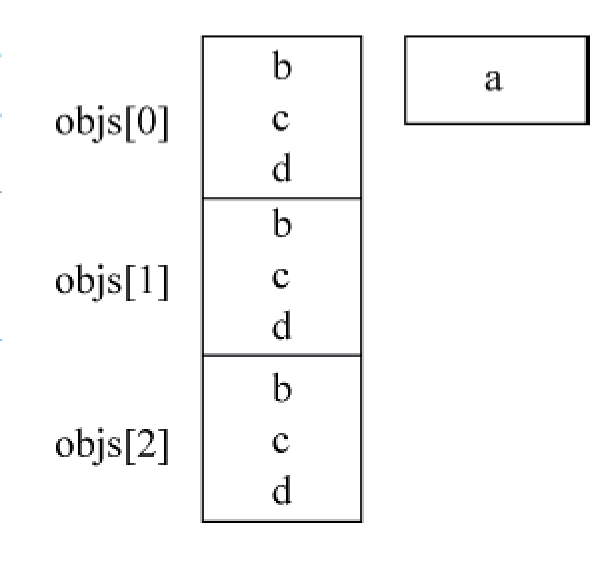


图 6-4 类 CS 的成员

还可以把一个成员函数说明为 static，这意味着在没有任何该类对象的情况下，它仍然可以被执行。正因如此，静态成员函数能够完成不需要任何对象成员变量的操作。静态成员变量和静态成员函数为类存储和管理属于整个类而不是个别对象的信息提供了一种方法。

5. 构造函数

在面向对象技术中，对象是类的实例，每个对象必须按照类的定义来创建，这种机制是通过类的构造函数来实现的。构造函数（constructor）是一种特殊的成员函数，用来在内存中建立具体的对象。构造函数必须申请必要的内存空间，将内存转化为具体的对象，初始化成员变量等。构造函数的名称和类名称相同，一个类可以拥有几个带有不同参数的构造函数。构造函数没有返回类型和返回值。

和一般的函数不同，构造函数不是由用户显式调用（call）的，它是通过编译器来调用的，称为激活（invoke）。当通过 new 创建一个类的对象时，相应的构造函数则被激活，来完成内存对象的初始化操作等。

在使用构造函数时，需要注意以下若干情况。

（1）构造函数不能描述为 const 和 volatile。

（2）构造函数不能是 static，因为构造函数需要初始化类的成员变量，但静态构造函数不能访问成员变量。

（3）构造函数不能被继承。当一个没有构造函数的类从一个含有构造函数的类派生时，将写它自己的构造函数。

（4）构造函数不能有返回值，也不可以是 void。

（5）定义一个类时，必须明确定义除缺省构造函数和拷贝构造函数以外的所有其他构造函数。缺省构造函数是不含有参数的构造函数，可以省略不写。

6. 类的继承性与派生类

从现存的对象出发建立一种新的对象类型，使它继承原对象的特点和功能，这就是对象的继承性。继承是许多层次对象的自然描述。从现存类派生出新类时，现存类称为基类

(base class),派生出的新类称为派生类。派生类可以对基类作如下变化。

(1) 增加新的成员变量。

(2) 增加新的成员函数。

(3) 重新定义已有的成员函数,即:子类可以对父类的方法进行覆盖(overriding)或重载(overloading)。所谓覆盖,是指在子类中定义了与父类中同名的函数,这时将根据当前的对象类型执行子类中的代码,即父类中的代码被覆盖。所谓方法重载是类中含有同名,但特征参数(参数数量、参数类型)不同的方法,编译时通过形式参数确定要调用的方法。

覆盖或重载反映了面向对象中函数的多态性,重载是一种编译时多态技术,因为程序在编译时可以根据特征参数决定要连接哪个同名函数。覆盖则是一种运行时多态,因为,具体运行基类还是子类的函数是根据运行时对象所属的类决定的,因此是一种运行时多态。

(4) 改变现有成员函数的属性。

派生类不能删除基类的成员变量和成员函数,实际上派生类往往是基类的扩充,是一种具体化和完善的过程。

类的派生创建了一个类族,派生类的对象也是基类的一个对象,它可用在基类对象可被使用的任何地方。可用多态成员函数来调整这种关系,以使得派生类在某些地方和它的基类一致,而在另外一些方面表现出其自身的行为特征。

类的派生是一种演化过程,即通过扩展、更改和特殊化,从一个已知类出发来建立一个新的类。类的派生建立了一个具有共同关键特性的类族,从而实现代码的重用。假设从一个已知基类 A 建立一个派生类 B,一般形式为:

```
class B  extends  A
{
    // 派生类 B 的成员说明
};
```

读作“类 B 由 A 派生”,它告诉编译器类 B 是一种 A,对基类 A 所作的修改和添加在括号内给出。在派生类对象中,编译器在内存中总是先放入基类的成员,后面是派生类独有的成员,如图 6-5 所示。

对于一个派生类,对于其父类中的成员,有特定的成员的访问规则,见表 6-1。

基类成员
派生类 特有成员

图 6-5 派生类对象内存结构

表 6-1 基类和派生类中成员的访问特性

基类 class A { … }	派生类 class B extends A { … }
private 成员	不可访问
protected 成员	保护
public 成员	公有
不可访问成员	不可访问

所谓不可访问成员,是一个派生类中的概念。一个根类(不是从其他类派生出来的类)中不存在不可访问的成员。但是在派生类中,不可访问是存在的,如根类中的 private 成员,就构成派生类的不可访问成员。如果再从派生类派生新的类,则派生类的不可访问成员和私有成员就构成了它的派生类的不可访问成员,依此类推。

最后,需要说明的是,在调试 Java 程序时,经常是编译通过了,但运行时出现下列错误

提示：Exception in thread "main" java.lang.NoSuchMethodError：main。产生上述错误的原因比较复杂，可以从以下两个方面进行排查。

（1）每个 Java 文件有且只能有一个公有类，即 public 类，文件名必须和这个公有类的类名大小写完全一样。在要运行的类中有且只能有一个 public static void main(String[] args)方法，这个方法就是主程序。运行一个 Java 程序，应该是在 java 命令后跟包含 main 函数的类名，而不是 Java 程序的文件名，即 java <包含 main 方法的 java 类>，类名后面不能有.class 等扩展名。

（2）如果在编译或运行时出现问题，还可能是类的路径问题。我们来看一下 Java 程序的运行过程，首先通过 javac 命令，将 Java 程序编译生成.class 文件，即虚拟机要执行的代码，称之为字节码(bytecode)。然后，通过 java 命令，即通过 Java 虚拟机来解释运行 class 文件。虚拟机通过 classloader 来装载这些字节码，即通常意义上的类。问题是，classloader 从哪里知道 Java 本身的类库及用户自己的类在什么地方呢？要解决这个问题，通常是通过系统环境变量中的类路径(classpath)来设置的。

因此，当出现编译或运行错误时，还应该检查类路径设置是否正确。具体方法是，在 DOS 提示符下输入 set classpath，即可显示当前的路径设置，应该包含一个当前路径项目，即"."，其次就是 Java 的安装路径。

7. 创建对象

当定义类后，创建对象有两种方法，一种是静态的声明，例如：CTest myObj；则创建一个 CTest 类的对象 myObj。此外，还可使用 new 来动态地创建对象，一般形式是：

```
对象变量 = new 类(参数);
```

如果类的构造函数带有参数，可以通过 new 中类名后面的参数，来激活相应的构造函数，来实现对类成员的初始化。如果是一个派生类，还涉及基类成员的初始化问题。

8. Java 程序与 main()方法

所有的程序都应该有一个主程序，它是程序开始执行的入口。在 Java 程序中，通常定义多个类，但只能有一个 public 类。在这些类中必须有一个类包含 main()方法，如果包含 main 方法的类是公有类(用 public 修饰的类)，那么文件名必须是这个类的类名，如果 main 方法所在的类没有用 public 修饰，那么文件名可以任意命名，不一定要和任何类名一样。

【例 6-2】 定义一个三角形图形类 CTriangle，定义相应的属性和方法，并在 DOS 模式下对文件进行编译和运行。

分析：定义了一个三角形图形类 CTriangle，包含三个私有成员变量，存储三角形的三条边，一个构造函数，一个计算三角形面积的成员函数和一个输出成员函数。

代码清单：Exa6-02.java

```
class CTriangle
{
  private double a,b,c,t;
  public double s = 0;
  CTriangle(double x,double y,double z)
```

```
    {
      a = x; b = y; c = z;
    }
    //计算三角形的面积
    public double Area()
    {
      t = (a + b + c)/2;
      s = Math.sqrt(t * (t - a) * (t - b) * (t - c));
      return s;
    }
    public void out1()
    {
      Area();
      System.out.println("Area = " + s);
    }
  }
  class MyTest01
  {
    public static void main(String[] args)
    {
      CTriangle myobj  = new CTriangle(10,20,15);
      myobj.out1();
    }
  }
```

在 DOS 提示符下，执行 javac Exa6-02.java 命令，编译，生成每个类的.class 文件，即 CTriangle.class 和 MyTest01.class。Java 程序是在 JVM 上运行的，在 DOS 提示符命令状态下，执行 java MyTest01，运行上述程序，输出结果如图 6-6 所示。

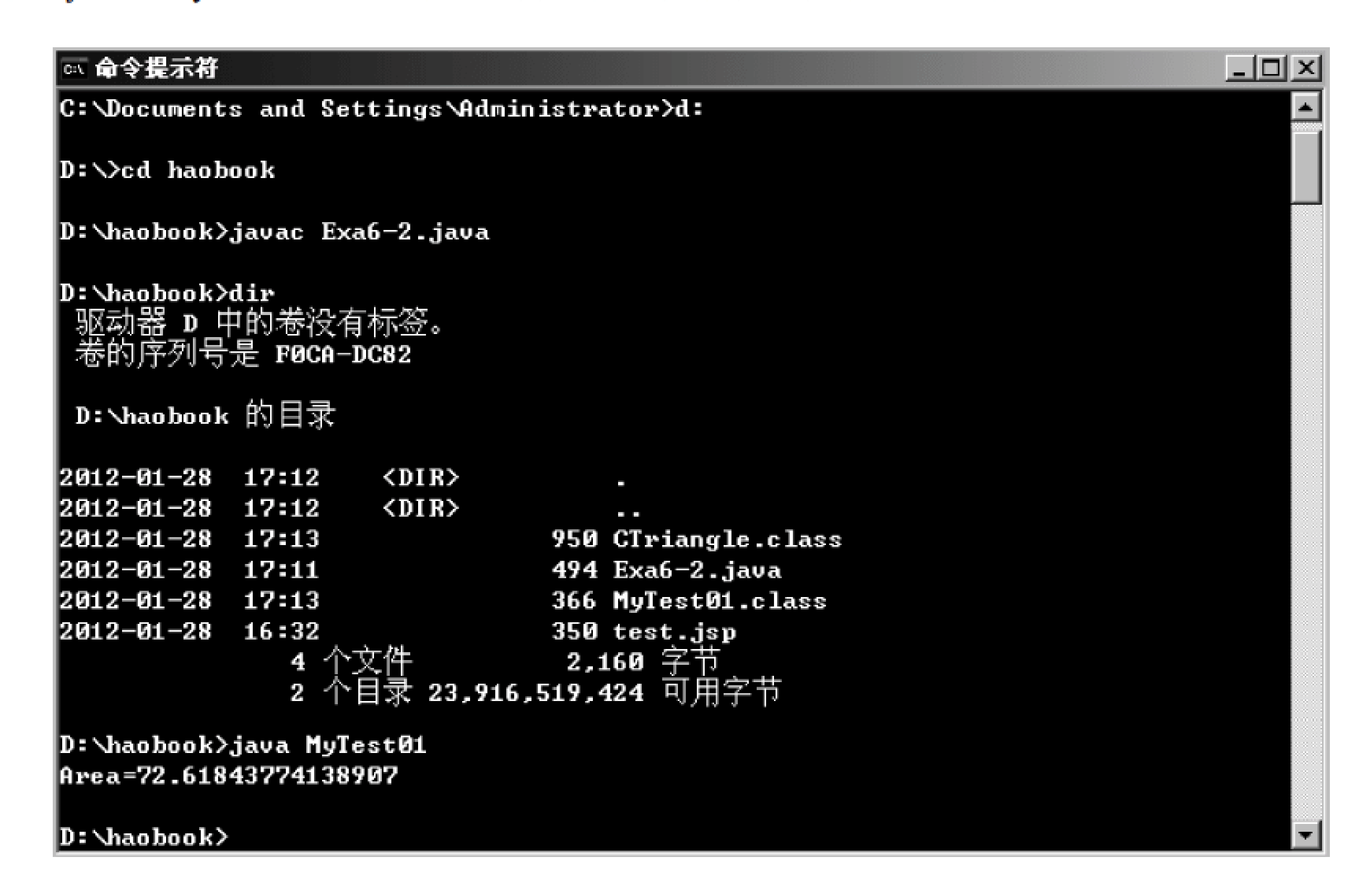


图 6-6　Java 程序的编译和运行

【例 6-3】　编写程序，验证类构造函数的激活顺序。

分析：在类的定义中，都包含相应的构造函数，这些构造函数的激活顺序不同，在下面的例子中，定义三个类 A、B、C，其中 C 中包含两个对象成员，来展示对象构造函数的激活

次序。

代码清单：Exa6-02.java。

```
class A {
   int x,y;
   A(int a,int b)
   {
     x = a; y = b;
   }
   A()  //默认构造函数
   {
     x = 0; y = 0;
     System.out.println("A constructor\n");
   }
}
class B {
   B()
   {
     System.out.println("B constructor\n");
   }
}
class C {
   public A a = new A();
   public B b = new B();
   C()
   {
     System.out.println("C constructor\n");
   }
}

class MyTest02
{
  public static void main(String[] args)
  {
    C myObj = new C();
  }
}
```

执行 java MyTest02，输出结果为：

```
A constructor
B constructor
C constructor
```

上述输出结果表明，如果一个类含有成员对象，则在创建类的对象时，将先创建类的成员对象，最后才是类的对象本身。

6.2.3　接口

在一个复杂的面向对象的系统中，实现一个有更多方法的新类是经常遇到的。当一个

类需要从多个基类派生时,派生类将继承多个基类的特征,在 C++中,这样的机制称为多重继承。在面向对象的程序设计中,继承关系一直存在很多的争议,特别是多重继承。

1. 接口和类

Java 没有多重继承,可以通过接口来实现相应的功能。在 Java 中,所谓接口(Interface)是一组没有给出实现细节的操作(方法)的集合。它需要别的类来实现接口给出的每一个方法,一个类可以实现一个或多个接口。如果一个类实现了某个接口,就相当于声明我能够完成某项工作。在许多情况下,类的实现接口继承(implements 关系)比类的扩展基类继承(extends 关系)更有优势。

接口的定义和类的定义类似,但是接口不是一个类,而是对符合接口要求的类的一套规范。接口说明了实现接口的类该做什么而不指定如何去做,一个类可以实现一个或多个接口。

实现一个接口需要两个步骤。

(1) 声明类需要实现的接口。声明一个类实现一个接口需要使用 implements 关键字。

(2) 提供接口中的所有方法的定义。为了使用接口,需要编写执行接口的类。一个类实现了某个接口,即这个类实现了在接口中声明的所有方法,就相当于声明我能够完成某项工作。

实现接口的类继承了定义在接口中的常量,这些类可以使用简单的名字来引用接口定义中的常量。

2. 接口的定义和扩展

在 Java 中,可以定义一个接口,也可以从一个接口或多个接口来扩展一个接口,这和类的定义类似。接口定义的一般形式是:

```
public interface < Interfacename > [extends < Superinterface 列表>]
{
  成员变量声明(常量);
  成员函数(方法)声明;
}
```

在接口定义中,public 指示了接口可以被任何包中的任何类使用。如果没有指定接口为 public,那么接口就只能在定义接口的包中的类中使用。一个接口可以扩展另外的接口,这跟类可以扩展一样。但是,类只能扩展一个另外的类,而接口可以扩展任意个接口。Superinterface 列表列出所有的被扩展的接口,以逗号分隔。

接口可以包含常量声明以及方法声明。所有定义在接口中的常量可以是 public、static 和 final。定义在接口中的成员变量不能使用 transient、volatile 或者 synchronized 修饰符。同样也不能在声明接口的成员的时候使用 private 和 protected 修饰符。

接口中的方法声明后紧跟着一个分号,因为接口中的方法不需要给出具体的实现代码。因此,所有定义在接口中的方法可以隐含地为 public abstact 方法。

6.2.4 包

在 Java 中,为了管理大型名字空间,避免名字冲突,引入包(Package)的概念。包是由

一组类和接口构成。一般情况下，每一个类或接口都被存储在不同的文件中，为了管理和使用方便，对于那些相关的类和接口可以绑定到一个包中。例如，Java 的基本类在 java. lang 中，而用于输入、输出以及文件(夹)操作的类则在 java. io 中。

Java 中的包其实指的就是目录，它可以更好地管理 Java 类(Class)和接口(Interface)。使用包就定义了一个类和接口的名字空间，一个包中的类和接口的名字与其他包中的名字不会冲突。同时，还可以约束包内的类和外部的类的访问权限。

1. 定义包

使用 package 语句可以将一个编译单元(源程序文件)定义成包。如果使用 package 语句，编译单元的第一行必须无空格，也无注释。格式如下：

```
package <包名>;
```

若编译单元无 package 语句，则该单元被置于一个默认的无名的包中。

按照一般的习惯，包名是由“.”号分隔的单词构成，第一个单词通常是开发这个包的组织的名称。

例如，有一个 Java 文件，文件名为 B. java，内容如下：

```
package hao.yy;
public class B {
    B(int yy, int mm, int dd)
    {
      System.out.println("Year:" + yy + "Month:" + mm + "Date" + dd);
    }
}
```

执行 javac -d d:\B. java 命令，其中参数-d ＜目录＞为指定输出文件的目录。因此，上述编译命令执行后将在“d:\”目录下，创建一个 d:\hao\yy 文件夹，该文件夹中包含了编译后的类文件 B. class。

现在这个包已经创建好了，要使用这个包中的类 B，需要导入，或者把 d:\hao\yy 设置在环境变量 classpath 中。

2. 使用包中的类和接口

要使用定义在一个包中的类和接口，可以通过 import 关键词，主要有两种形式。

(1) 使用 import 语句，导入类或接口，或包含它们的包。导入的类和接口的名字在当前的名字空间可用。导入一个包时，则该包中的所有的公有类和接口均可用。形式如下：

```
import <包名.[ * | 类名 | 接口名>;
```

其中，“*”表示导入包中的所有类。如果仅仅要使用包中一个类，可指定类或包的名字。例如：使用 util 包的 Date 类，可以写作“import java. util. Date;”，下面是一个有关日期处理的类的定义，程序获取当前的系统时钟，并按 MM-dd-yyyy 格式输出日期。代码如下：

```
import java.util.Date;
import java.text.SimpleDate;
```

```
public class mydate
{
    public static void main(String[] args) {
        // Create a date formatter that can parse dates of the form MM - dd - yyyy
        SimpleDateFormat bartDateFormat = new SimpleDateFormat("MM - dd - yyyy");
        // Create a string containing a text date to be parsed
        String dd = "6 - 22 - 2008";
        Date date = bartDateFormat.parse(dd);
        // Send the parsed date as a long value to the system output
        System.out.println(date.getTime());
    }
}
```

(2) 在每个引用的类和接口前面给出它们所在的包的名字。一般形式是:

```
包名.<类名|接口名> obj = new 包名.<类名|接口名>(参数表);
```

例如,要使用上面定义的包 hao.yy 中包含的类,语句为:

```
hao.yy fd = new hao.yy.B(2003,11,24);
```

在使用一个外部类或接口时,要声明该类或接口所在的包,否则会产生编译错误。此外,要确保这些类和包的路径正确,即需要包含在 classpath 环境变量中,否则编译器在编译时将找不到所需要的类。

6.2.5 Java 基础类库

对于所有的程序设计语言,都可以分成两个部分,第一部分就是语言本身,包括语言的字符集、数据、类型、程序语句、函数等语法部分;第二部分则是开发程序用的标准库,里面包含了大量的标准函数或标准类,是软件开发的 API(Application Program Interface),它可以帮助开发者方便、快捷地开发相应语言的程序。

在 Java 编程中,Java 语言提供了大量的已经实现的标准类,这些类的集合构成 Java 的类库。这些类根据实现的功能不同,划分为不同的集合,每个集合组成一个包。Java 类库大部分都是由 Sun 公司提供的,这些类库称为 Java 基础类库。Java 的类库日益庞大,所包含的类和接口众多,用户是无法全部掌握的。但是,了解类库的组成可以帮助开发者更方便地找到需要的类,节省大量的编程时间,使编写的程序更简单。Java 中丰富的类库资源也是 Java 语言的一大特色,是 Java 程序设计的基础。下面简要介绍 Java 类库中的核心部分,即那些包含最重要、最常用的类和接口的包。

1. java.lang 包

java.lang 包又称 Java 语言包,主要含有与语言相关的类。定义了 Java 中的大部分基本类,包含 Java 语言类、线程、异常、系统、Object 类以及各种数据类型等相关的类。java.lang 包是 Java 程序中默认加载的一个包,由解释程序自动加载,不需要显式说明。

2. java.util 包

java.util 包,又称 Java 实用程序包。Java 平台中有两个最常用的基础包,一个是 java.

lang 包,另一个就是 java. util 包。java. util 包包含了大量的公用类,包括常用的数学运算类、字符串类、日期、日历类以及向量哈希表等类,还包括一些接口和异常类。

3. java.awt 包

Java 抽象窗口工具(Abstract Windowing Toolkit)包 java. awt 包含一些 GUI 界面相关的类,包括窗口、对话框、菜单、各种控件等。awt 类库还包含一组用来处理绘图、打印功能,并且支持易用性、拖放和二维图形的 API。通过这些元素,编程者可以控制所写的 Applet 或应用程序的外观界面。

通过 awt 可以创建与平台无关、基于图形用户界面的程序。同微软的 Windows API 相比,清楚、简单和强大的 awt 是 Java 语言迅速流行的重要原因。awt 不仅是编写 Windows 程序的良好工具,而且可以编写其他操作系统平台的图形界面应用程序。

4. java.swing 包

新的图形界面类库 java. swing 继承 awt,提供了多种图形界面组件,swing 中包含了像标签页、表格、树、特殊边框、微调等各种新组件。这些组件都是 100%纯 Java 的,不依赖具体的 Windows 系统,可以在各种平台上实现。swing 中支持可插入观感(Pluggable Look and Feel,PL&F),支持用户定制桌面、更换新的颜色方案,让窗口系统适应特定的用户习惯和需要。swing PL&F 体系结构使得同时定制 swing 控件或控件组更加容易。

5. java.io 包

输入输出包 java. io,提供了全面的 I/O 接口,包括文件读写、标准设备输出等。它是以流为基础进行输入输出的,所有数据被串行化写入输出流,或者从输入流读入。I/O 体系分 Input/Output 和 Reader/Writer 两类,区别在于 Reader/Writer 在读写文本时能自动转换内码。流 I/O 的好处是简单易用,缺点是效率较低。

Java 也对块传输提供支持,在核心库 java. nio 中采用的便是块 I/O,块 I/O 效率很高,但编程比较复杂。

6. java.applet 包

java. applet 包中定义了设计小应用程序(Applet)的类和接口,包括控制 HTML 文档格式、应用程序中的声音等资源的类。例如 Applet 类、AppletContext 接口、AppletStub 接口、AudioClip 接口等,其中 Applet 是创建包含于 HTML 的 Applet 必不可少的。

7. java.beans 包

JavaBeans 是 Java 应用程序环境的中性平台组件结构。java. beans 包定义了应用程序编程接口(API),包含与开发 JavaBeans 有关的类和接口。

8. java.net 包

java. net 包包含有与网络操作相关的类,如 TCP Sockets、URL 等工具,该包支持 TCP/IP,并包含 Socket 类、URL、与 URL 相关的类。

6.2.6 Java Servlet 服务器程序

在一台 Web 服务器控制下,运行若干小型用户程序,这就是公共网关接口(CGI)程序(又称 CGI 脚本)所起的作用。CGI 应用程序本身往往不是完整的应用程序,在处理接收自 Web 浏览器上用户的信息请求时,CGI 只是整个处理过程中的一个中间步骤。例如,CGI 应用程序的一种常见用途是访问数据库。将它用于这种任务时,CGI 程序提供一种方法,将用户的数据请求连接到能满足这种请求的企业数据库。CGI 程序常常充当一种中间软件,从 Web 浏览器接收请求,决定必须调用哪些计算资源来满足这些请求,并向浏览器发回响应。传统的 CGI 程序一般是由 C 或 C++开发的,这些语言的执行速度较快。随着优化编译器的引入,用 Java 来实现中间层成为可能。

Servlet 是专门为在 Web 服务器上运行而设计的 Java 程序,与 CGI 程序一样,用于充当连接前端 Web 请求与后端数据资源的中间层组件。它可以动态地扩展服务器的能力,并采用请求-响应模式提供 Web 服务。使用 Java Servlet 可以以更高的效率和可移植性来实现 CGI 的目的,并最终会取代传统的 CGI 程序。

现在,在 Web 开发中,传统的 CGI 编程正在被 Servlet 技术所替代。Java Servlet 的出现,为应用程序员使用 Java 来创建 Web 应用程序开辟了新的途径。但是,Servlet 只是工作在 Web 服务器上的一个连接客户请求和数据库系统的中间层,它不是一个可以独立运行的 Java 程序,它不能在操作系统下直接运行,必须运行在 Servlet 容器中。

单独编写 Servlet 非常麻烦,随着 JSP 技术的发展,可以直接将 Java 代码写在 HTML 页面中,这些代码被包含在“<%”和“%>”内,形成 JSP 文件。对于 JSP 页面中的代码,将被编译成 Servlet,这样就大大地减少了 Servlet 开发的难度。

6.3 JavaBean

在软件开发中,软件组件技术的思路就是把软件功能开发为一个个的组件,将不同的组件组装成一个应用程序,而 JavaBean 组件是软件组件技术的一种 Java 实现。在 Web 应用开发中,JavaBean 通常用做中间“商业逻辑”层,用于将表示逻辑和数据访问逻辑隔离,许多业务逻辑通常被编码成 JavaBean,而不是直接写在 JSP 页面中。由于 CGI/Servlet 的复杂性,目前 JavaBean 和 JSP 技术已经成为 Web 应用服务器三层结构中的中间层业务逻辑开发的主要工具。

6.3.1 JavaBean 的概念

简单地讲,JavaBean 是一种 Java 类。任何具有某种特性和事件接口约定的 Java 类都可以写成一个 JavaBean。通过将一个满足某些准则的 Java 类写成 JavaBean,通常用于 JSP 页面中,以便于实施软件的重用。

理论上讲,虽然 JavaBean 没有严格的规范,通常情况下,一个 JavaBean 是一个 Java 类,这样的 Java 类将对应于一个独立的.java 文件,在绝大多数情况下,它是一个 public 类型的类。作为一个类,它具有类的一般特征,例如属性、方法等。

1. 属性

JavaBean 提供了高层的属性(Properities)概念。从面向对象的角度看,属性就是传统的对象属性(Attribute)。JavaBean 属性描述了 bean 的内部状态,是 JavaBean 中的数据部分,属性的值可以通过适当的方法进行读写。JavaBean 有四种类型的属性,分别是单值属性、索引属性、关联属性和限制属性。

1) 单值属性

单值属性是 JavaBean 中最简单的属性,只需要定义一个包含一个值的数据成员,并为其定义一对设置(set)/获取(get)属性的方法,以便于外部与其发生联系。如果没有为单值属性提供设置器(set)方法,则该属性为只读型属性;如果没有为单值属性提供获取器(get)方法,则该属性为只写属性。

单值属性设置器/获取器定义的一般形式是:

```
public void set<PropertyName>(<PropertyType> propertyValue)     //属性设置器
public <PropertyType> get<PropertyName>()                        //属性获取器
```

每一个单值属性都有一组 set 和 get 方法用于给属性赋值或获取该属性的值。

2) 索引属性

索引属性类似于 Java 中的数组,包括若干个数据类型相同的元素,可以通过整数索引值访问其中的属性,因此称为索引属性。

3) 关联属性

JavaBean API 除了支持单值属性和索引属性外,还提供了一些属性用于增强 JavaBean 的属性管理功能。如关联属性,当修改这类属性时,将发送一个通知给其他元素(如 Applet、Application 或其他 JavaBean),如果与该 JavaBean 中的某个属性相关联,就会注册该属性。因此,只要这个关联属性发生变化,就会有一个通知发送给这些相关部件,这些属性称为关联属性,它们的值发生改变与外部部件有关,外部部件称为监听器。

一个有关联属性的 JavaBean 需要支持如下一对事件监听器的注册方法:

```
public void addPropertyChangeListner(PropertyChangeListner1)
public void removePropertyChangeListner(PropertyChangeListner1)
```

4) 限制属性

JavaBean API 中的另一种高级属性类型是限制(Constrained)属性,它可以使外部部件在接受属性的修改值之前先确认修改值。也就是说,当需要修改一个限制属性值时,接受属性的外部部件首先要检查这个属性的合理性再决定是否接受修改。

2. 方法

JavaBean 归根到底是一个 Java 类,所有的 public 方法都可以通过对象外部直接调用。JavaBean 中常用的方法是单值属性设置器/获取器,即对于每一个属性,都有一对 set 方法和 get 方法来实现对属性值的赋值和获取属性值。

3. 事件

JavaBean 和其他软件组件交流信息的主要手段是发送和接收事件。在新的 AWT 事

件模型中,一个事件源可以注册事件监听器对象,当事件源检测到某种事件发生时,将激活检测器对象中一个相应的事件处理方法。

JavaBean 必须能够发送事件变化通知和监听到其他 JavaBean 的事件变化,并对监听到的事件变化进行相应的业务处理。事件的监听原理是:首先事件源必须对需要发送的事件进行注册,然后注册事件监听器,并说明该事件源所发生的事件向什么组件发送,也就是说,在事件源组件中实现方法并在监听组件中注册该事件源。

【例 6-4】 JavaBean 应用举例。

在 Web 应用中,用户定义的类或 JavaBean 通常存储在根目录下的"WEB-INF\classes\包\"文件夹中,Java 通过包来实现用户类的分类存储和管理。

一个名为 NameCard 的 JavaBean 代码如下(文件名 NameCard.java):

```
package cards;
public class NameCard
{
   String Name,Address;
   public NameCard()        // 默认构造函数
   {
     this.Name = "John";
     this.Address = "No,Road,City,Country";
   }
   public void setName(String myName)
   {
     this.Name = myName;
   }
   public String getName()
   {
     return (this.Name);
   }
   public void setAddress (String myAddress)
   {
       this.Address = myAddress;
   }
   public String getAddress ()
   {
       return (this.Address);
   }
}
```

上述 JavaBean 应存储在 Web 应用特定的文件夹中,即 Web 应用根目录中"WEB-INF\classes\cards"下,用 javac NameCard.java 编译生成 NameCard.class,该文件将存储在 cards 包中,即 cards 子文件夹中。

当 JavaBean 完成后,为了测试 JavaBean 的功能,可以在 bean 类中临时增加 main 方法,借用 main 方法来调试 bean。JavaBean 的主要应用是将 JavaBean 应用于 JSP 页面中,关于在 JSP 中使用 JavaBean,将在 6.4.5 节进行详细介绍。

6.3.2 JavaBean、EJB 和 Java 类的区别

JavaBean 是可复用的组件,对 JavaBean 并没有严格的规范,理论上讲,任何一个 Java

类都可以是一个 Bean。但通常情况下，由于 JavaBean 是被容器所创建（如 Tomcat）的，所以 JavaBean 应具有一个无参构造函数。JavaBean 都是 public 类，对于一个名为 x 的属性，通常要写两个函数 getX()和 setX()，对于其 Bean 的一般函数不需要遵守上述的命名规则，但是需要为 public。此外，通常 JavaBean 还要实现 Serializable 接口用于实现 Bean 的持久性，JavaBean 不能被跨进程访问。

Enterprise JavaBean(EJB)是基于 Java 的远程方法调用(RMI)技术，所以 EJB 可以被远程访问(跨进程，跨计算机)。但 EJB 必须被部署在诸如 WebShpere、WebLogic 这样的容器中，EJB 客户从不直接访问真正的 EJB 组件，而是通过其容器访问。EJB 容器是 EJB 组件的代理，EJB 组件由容器所创建和管理。客户通过容器来访问真正的 EJB 组件。

6.4　JSP 技术

使用 Servlet 开发服务端中间层逻辑，实在是太复杂了。为此，Sun 公司于 1999 年推出 JSP 技术。JSP 是一种 Java 平台技术，它是在传统的 HTML 文档(*.htm、*.html)中加入 Java 脚本程序(Scriptlet)和 JSP 标记来构成的，它包装了 Java Servlet 系统界面，用户可以在 JSP 页面上直接书写 Java 代码，这样就大大简化了 Java 和 Servlet 的使用难度。此外，作为一种服务器脚本语言，JSP 是在编译成 Servlet 后才能实际运行。当用户浏览.jsp 网页时，首先在服务器端执行.jsp 文档中服务端代码，然后把执行结果传送到客户端浏览器，基本上与浏览器无关。

6.4.1　JSP 的运行环境

要运行 JSP 页面，在 Web 服务器端，需要安装 Tomcat 应用服务器。Tomcat 通常和 Apache Web 服务器一起共同构成一个 Web 站点。

1. 运行与开发环境

JSP 的运行和开发环境框架模型如图 6-7 所示。

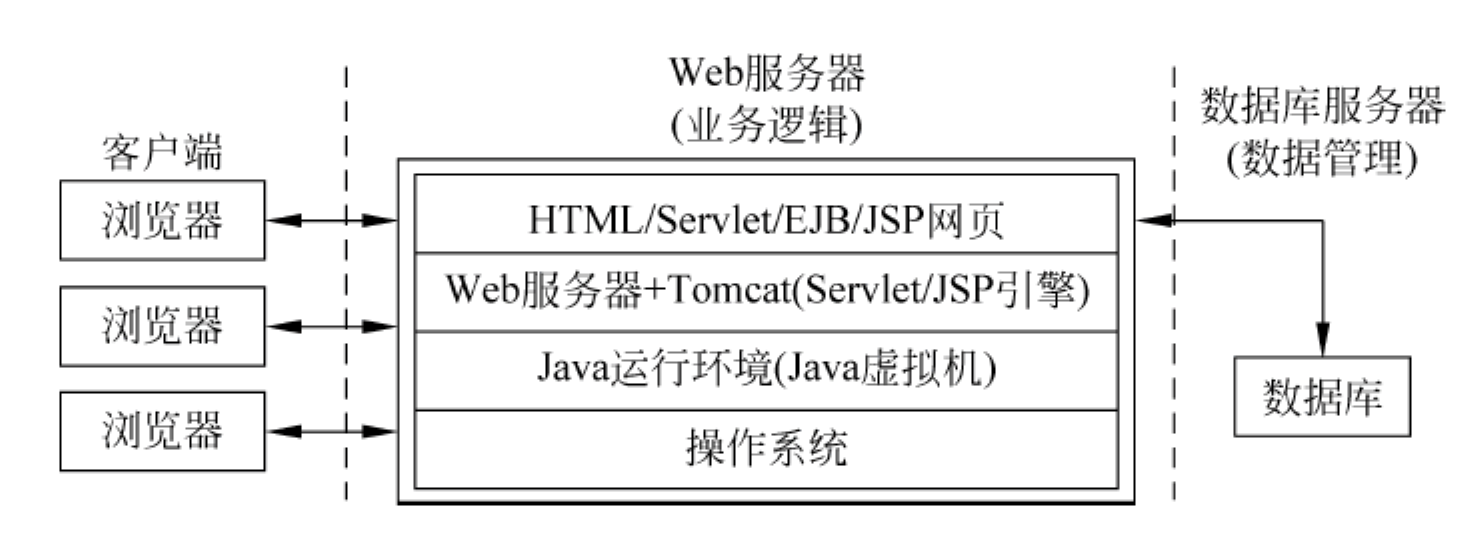


图 6-7　JSP 的运行和开发环境

JSP 的开发、运行环境很多，可以采用 UNIX、Linux、Windows Server 等不同类型的操作系统，如果希望在服务器上开发 Servlet/EJB/JSP 应用程序，还应该安装以下软件。

(1) 安装 Java 环境。

(2) 安装 Java VM(JRE)，Tomcat 需要 Java VM 的支持。

(3) 安装 Tomcat 应用服务器。Tomcat 是针对 Apache 服务器开发的 JSP 应用服务器,是 Java Servlet 和 Java Server Pages 技术的标准实现,是基于 Apache 许可证开发的自由软件。可以这样认为,当在一台机器上配置好 Apache 服务器,可利用它响应对 HTML 页面的访问请求。实际上 Tomcat 部分是 Apache 服务器的扩展,但它是独立运行的,所以当运行 Tomcat 时,它实际上作为一个与 Apache 独立的进程单独运行。当配置正确时,Apache 为.html 页面服务,而 Tomcat 实际上运行.jsp 页面和 Servlet。

(4) Apache 服务器。可以根据需要安装 Apache 服务器。本章主要是介绍 JSP 的开发,因此,只要安装 Tomcat 就可以了。

2. JSP 程序页面测试站点

为了测试后面要学习的 JSP 程序,我们继续使用 6.1.2 节建立的站点 d:\haobook,虚拟目录为"/book"。所有的要测试的 JSP 页面,只要保存到 d:\haobook 文件夹中即可。如果要测试一个 JSP 文档 1.jsp,在地址栏中输入 http://127.0.0.1/book/1.jsp。注意,必须要输入扩展名,同时要注意文件名和目录的大小写。

6.4.2 JSP 的语法结构

JSP 文档是通过在 HTML 文档中加入 Java 脚本程序构成的。Java 脚本程序代码用"<%"和"%>"定界符括起来,称为 JSP 元素。JSP 元素可以分为三种类型:脚本元素、指令元素和动作元素。脚本元素规范 JSP 中所使用的 Java 代码;指令元素针对 JSP 引擎控制转译后的 Servlet 的整个结构;动作元素主要连接用到的组件(如 JavaBean 和 Plugin),另外它还可以控制 JSP 引擎的行为。

1. JSP 指令

JSP 指令是 JSP 的引擎。它们不直接产生任何可视的输出,只是指示引擎如何处理 JSP 页面中的内容。JSP 指令由<%@ …%>标记,一般形式是:

```
<%@ 指令名 属性1="属性值" 属性2="属性值" …属性n="属性值" %>
```

在指令书写时,指令名和"@"符号之间需留有空格,常用的 JSP 指令是 page 指令和 include 指令。

1) page 指令

page 指令(at page)用来定义整个页面的相关属性,以及定义网页的处理方式,如到何处寻找 Java 类支持等。指令的语法如下:

```
<%@ page
    [ language="java" ]
    [ extends="package.class" ]
    [ import="{package.class | package.*}, …" ]
    [ session="true | false" ]
    [ buffer="none | 8kb" ]
    [ autoFlush="true | false" ]
    [ isThreadSafe="true | false" ]
```

```
[ info = "text" ]
[ contentType = "mimeType [ ;charset = characterSet ]" |
                 "text/html ; charset = ISO - 8859 - 1" ]
[ errorPage = "relativeURL" ]
[ isErrorPage = "true | false" ] %>
```

下面是 page 指令的几个常用属性。

(1) language 属性,所使用的脚本语言。例如,<%@ page language="Java" %>。

(2) import 属性,脚本元素中使用的类,与 Java 中的 import 声明作用相同,应是类的全名,或者类所在的包。例如:

```
<%@ page import = "java.util.Date" %>
<%@ page import = "java.io.*" %>(java.io 包中的所有类在本页中都可以使用)
```

(3) session 属性,是否使用 session 对象。

(4) buffer 属性,对象 out 的输出模式。none 为没有缓冲区;8KB 为缓冲区大小。

(5) autoFlush 属性,缓冲区已满时是否自动清空。当 buffer 为 none 时该属性不能为 false。

(6) isThreadSafe 属性,处理对象间的存取是否引入 Thread Safe 机制。如果为 true,则在程序中必须有多线程的程序代码,否则直接实现 SingleThreadModel 机制。

(7) contentType 属性,输出到客户端的 MIME 类型和字符集。默认的字符集是 ISO-8859-1。例如,使用中文字符集:<%@ page contentType=" text/html;gb2312" %>。

(8) errorPage 属性,设置异常处理网页。

(9) isErrorPage 属性,当前网页是否为另一个 JSP 网页的异常处理网页。

在使用 page 指令时,如果指令的属性较多,可以写成多条 page 指令。几乎所有的 JSP 页面都会用到 page 指令,在 JSP 文档的开始,常常看到如下的 page 指令,以定义网页的处理方式:

```
<%@ page import = "java.util.Date" %>  //导入页面中用到的 java 类
<%@ page errorPage = "errorPage.jsp" %>//当出现错误时的错误处理网页
<%@ page session = "true" %>
```

2) include 指令

include 是一个在 JSP 网页中包含其他文件的指令,它是在 JSP 引擎将它转译成 Servlet 时产生作用的指令。其一般形式如下:

```
<%@ include file = "被插入文件的 url" %>
```

这里要求,被包含的文件必须符合 JSP 的语法,应是静态文本、指令元素、脚本元素和动作元素。注意,包含后面不能有分号。

2. 变量声明

在 JSP 中,变量同样有全局变量和局部变量的概念。如果一个页面中包含多个"<%"和"%>"定界符括起来的 JSP 元素,在一个元素内部声明的变量只能用于该 JSP 元素内部。如果希望变量声明可以用于页面中的所有 JSP 元素,则需要定义页面级的变量。变量

声明一般形式如下：

```
<%!
类名 变量名[,变量名][,变量名]…;
…
%>
```

在变量声明中，可以为变量赋初值，需要用分号来结束变量声明，同时任何内容必须是有效的 Java 语句。例如：

```
<%!
String truename,nickname;
String[] ss;
int i = 0;
java.util.Date newsdate;
%>
```

注意，声明后面的分号不能省略。

3. 表达式

表达式是常量、变量、函数、运算符、括号连接而成的式子。在程序设计中，通过表达式完成对数据的运算，以及为变量赋值。在 JSP 文档中，可以将表达式的结果直接输出到页面中。一般形式是：<%=表达式%>。例如：

```
<% = i %>           //输出变量 i 的值
<% = "Hello" %>     //输出字符串常量
```

需要特别注意的是“%”和“=”之间不能有空格。

4. 代码段/脚本片段

JSP 代码段或脚本片段包含在“<% … %>”标记对内。当 Web 服务器响应请求时，这种 Java 代码就会运行。在脚本片段周围可能是纯粹的 HTML 或 XML 代码，在这些地方，代码片段可以创建条件执行代码，或只是调用另外一段代码。

例如，以下的代码组合使用表达式和脚本片段，分别按照 H1、H2、H3、H4 和 H5 标题样式，显示字符串“你好”，脚本片段并不局限于一行源代码中：

```
<% for (inti = 1; i <= 4; i++) { %>
<H<% = i %>>你好</H<% = i %>>
<% } %>
```

上述代码在服务端执行后，发送到客户端的 HTML 代码为：

```
<H1>你好</H1><H2>你好</H2><H3>你好</H3><H4>你好</H4>
```

5. 注释

在文档中加入 HTML 注释，用户可以通过查看页面源代码来看到这些注释的内容。如果不想让用户看到注释内容，应将其嵌入<%-- … --%>标记对。一般形式是：

```
<% --注释内容-- %>
```

【例 6-5】 编写一个 JSP 文档，显示网页的访问次数。

首先定义一个统计访问次数的文档，文档名为 mycount.jsp。内容如下：

```
<%! private int accessCount = 0; %>
<table border = "0" width = "100%" height = "60" bgcolor = "#FFFF00">
  <tr height = "50">
    <td width = "20%" >主机名：<% = request.getRemoteHost() %></td>
    <td width = "20%">访问次数：<% = ++accessCount %></td>
    <td>当前时间：<% = new Date() %></td>
  </tr>
</table>
```

定义一个 JSP 页面，包含上述文件，输出一个随机数。mypage.jsp 文档内容如下：

```
<%@ page import = "java.util.*" %>
<html>
<body>
<%! Random RdmNumber = new Random(); %>
<%@ include file = "mycount.jsp" %>
<%
   out.println(RdmNumber.nextInt(100));   //输出 100 以内的随机整数
 %>
</body>
</html>
```

将上述文件保存在 d:/haobook 文件夹中，打开浏览器，在浏览器地址栏中输入 http://127.0.0.1/book/mypage.jsp 可以看到网页的输出结果。

6.4.3 JSP 中的数据类型及其转换

数据类型是一个编程环境中非常重要的语言要素，在许多情况下需要进行类型的转换。首先，不同类型的数据的运算不同，因此，还经常需要将不同类型的数据进行转换，例如将字符串类型转变为数值型等。其次，如果要将数据插入数据库中，为了满足数据库字段类型的需要，通常也需要对 JSP 页面中的数据类型进行转换。我们不准备详细地介绍 JSP 的所有数据类型，只是对常用的数据类型及其转换进行简要介绍。

1. 基本数据类型与包装类

在 JSP 中，数据类型分为基本数据类型和类两种形式，基本数据类型有：①int，按照长度不同，又分为 byte(8b)、short(16b)、int(32b)、long(64b)四种；②float，分为 float(单精度，32b)和 double(双精度，64b)；③char，unicode 字符；④boolean，变量的取值有 true、false。与上述基本数据类型对应的是对应的类，分别是 Integer、Short、Byte、Long、Float、Double、Character、Boolean，又称包装类。

将一个基本类型数据转换为类对象，称为正向转换。正向转换可以通过 new(一个新的类对象)调用构造函数完成，例如：Integer a= new Integer(2)；float b =new Float(3.14)。

将一个类对象转换为基本类型数据，称为反向转换。这可以调用类的成员函数实现。

例如：int ia=a.intValue();其中 a 是一个 Integer 对象。float fb=b.floatValue(),b 为 Float 对象。

2. 整数类型和字符串类型的转换

在 JSP 中,int 和 String 数据类型间互相转换的具体方法如下。

(1) 将字串 String 类型转换成整数 int 类型。通过 Integer 类可以将字符串转化为某种进制的整数数据,一般形式是：

```
int i = Integer.parseInt([String]);
```

或

```
int i = Integer.parseInt([String],[int radix]);
```

或

```
int i = Integer.valueOf(my_str).intValue();
```

其中,最后一种转换是将一个字符串转化成一个 Integer 对象,然后再调用这个对象的 intValue()方法返回其对应的 int 数值。

(2) 将整数 int 类型数据转换成字符串 String 类型数据。

一般形式是：

```
String s = String.valueOf(i);
```

或

```
String s = Integer.toString(i);
```

或

```
String s = "" + i;
```

对于 Double、Float、Long 类型数据和字符串类型之间的转换方法大同小异。例如：

```
long ln = java.lang.Long.parseFloat("123.5");
```

或

```
float f = Float.valueOf("123.5").floatValue()
```

详细介绍略。

3. 字符串类型和日期型数据的转换

在数据库中,通常有 Datetime 类型的数据字段。在 MySQL 或 MS SQL 中,可以直接把字符串插入日期类型的列中,SQL 会隐式做格式转换,将字符串类型转为日期类型。但是,字符串的格式必须是 yyyy-mm-dd 或 yyyymmdd 形式,例如 2008-12-10 或 20081210。

如果从数据库中读取一个 Datetime 类型的数据字段 NewsDate,可以有不同的读取方式,包括 getString("NewsDate")、getDate("NewsDate")等,前者返回一个 String 数据,后者返回一个 Date()类型的数据。

在 JSP 中，也可以将字符串转换成 Date 类型。例如：

```
String strDate = '03/16/2012';
java.util.Date mydate = new SimpleDateFormat("dd/MM/yyyy").parse(strDate );
```

即可得到对应的日期数据。

【例 6-6】 编写一段 JSP 代码，从 SQL 数据库读取一系列的新闻记录，显示新闻列表。如果新闻发布日期未超过 7 天，在新闻列表项后显示红色的 New。

分析：这是网站中一个常见的功能，主要技术点就是数据库中 Datetime 数据的读取以及 Date 类型的操作。关于数据库的读取，使用 SQL 命令，语法容易理解。

代码清单：

```
<%
String newsid,newstitle,newsdatestr,newstr;
java.util.Date newsdate,dNow = new java.util.Date();
long daydiff = 0;
strSQL = "SELECT NewsID,NewsTitle,NewsDate "
        + "FROM news "
        + "WHERE NewsType = N'" + newstype + "'"
        + "AND NewState = 1 "
        + "ORDER BY NewsDate DESC ";
rs = gslpub.executeQuery(strSQL);
int i = 0;
while (rs.next())
{
    newsid = rs.getString("NewsID");
    newstitle = rs.getString("NewsTitle");
    // 数据库中的日期型数据,也可以是 getString
    newsdatestr = rs.getString("NewsDate");
    newsdate = rs.getDate("NewsDate");
    //SimpleDateFormat formatter = new SimpleDateFormat("yyyy.MM.dd");
    daydiff = (dNow.getTime() - newsdate.getTime())/(24 * 60 * 60 * 1000);
     if (daydiff <= 7)
        newstr = "<font color = 'red'>[<i>new</i>]</font>";
     else
        newstr = "";
    newstitle = "<a href = 'mynews/news - page.jsp?newsid = " + newsid + "&closeflag = 1'" +
newstitle + "(" + newsdatestr.substring(0,10) + ")</a>" + newstr;
     out.print("<p class = 'text01'>" + newstitle + "</p>");
}
%>
```

如果通过 import java.util.Date;导入 Date 类，则在类对象声明前无须指定 java.util.Date，直接使用 Date <对象>即可声明一个 Date 对象。

4. 字符串类型和字符串数组类型的转换

在 JSP 页面中，一种常用的操作就是将一个字符串 split 转换为一个字符串数组，或将一个字符串数组合并成一个字符串。例如，将 string="aa,bb,cc"转换成 Vector 数据类型，分别包含三个字符串元素——"aa"、"bb"、"cc"。代码如下：

```
String strData = " aa,bb,cc ";
```

```
String strlist[] = new String[20];
strlist = strData.split(",");
```

上述操作可以间接地实现对字符串数组赋初值。

在 Web 开发中,用做字符串分隔符的字符可能是字符串本身数据的一个字符,因此在实际编程时,如何选取字符串分割符非常重要。根据我们的开发经验,通常使用两个不可打印的字符"\10"和"\20"作为分割符,因为这两个字符用户很少直接输入,不太可能是字符串本身的数据。例如,在下面的示例代码中,字符串 ss 中,定义了 2 级字符串的分隔结构,分别存储了一组超链接的显示文本和文件路径。

```
<%
String f1 = "\10";
String f2 = "\20";
String ss;
String[] s1,s2;
if (ss.length()> 0)
{
  s1 = ss.split(f1);
  for(int k = 0;k < s1.length;k++)
  {
     s2 = s1[k].split(f2);
     out.print("<a href = '" + s2[1] + "'>" + s2[0] + "</a><br>");
  }
}
```

注意,在上述代码中,求字符串的长度和字符数组的长度使用的方法不同。如果最后一个分割符后面没有内容,即字符串 ss 的最后一个字符是分隔符 f1,则分割后的数组将不会包含一个空元素。

对于字符串对象的操作,在编程时注意以下两个常用操作的使用。①equals()方法,字符串比较运算,使用时一般需要将字符串常量放在前面,变量放在方法内,这样当变量值为 null 时不会产生异常。例如:"a".equals(str),str 为 null 时,不会报错。执行 str.equals("a"),当 str 为 null 时,程序将发生异常而报错。②trim(),字符串截尾函数,对于一个取值为 null 的 String 对象进行 trim()操作,将发生异常,输出 null,因此在操作前应该使用 if(str!=null) str=str.trim()。

5. 数组类型和集合类

无论是什么样的程序设计,数组(Array)都是一种常用的数据类型。和上述简单的数组类型相比,Java 中还提供了 Vector、ArrayList 集合类。集合类和数组不同,当数组中元素的个数不确定时,可以使用 java.util.Vector 类。例如:

```
Vector v = new Vector();
for (int i = 0; i< strlist.length;i++)
    v.add (strlist[i]);
return v;
```

和 Array 数组相比,Vector 集合只能存放 java.lang.object 对象,不能用于存放基本类型数据。例如,要存放一个整数 10,得用 new Integer(10)构造出一个 Integer 包装类对象,

才能 add 到 Vector 集合中。和 Array 相同的是也通过元素的整数索引来访问 Vector 元素,调用 Vector 的 size()方法时,可以返回 Vector 集合中实际元素的个数。

在 Java 中, Vector 类和 ArrayList 类具有相似的功能。从内部实现机制来讲, ArrayList 和 Vector 都是使用数组来控制集合中的对象,当从一个指定的位置(通过索引)查找数据或是在集合的末尾增加、移除一个元素时,两者所花费的时间相同。

但是,从同步的角度讲,Vector 是同步的,它的一些方法保证了 Vector 中的对象是线程安全的。ArrayList 则是异步的,因此 ArrayList 中的对象并不是线程安全的。因为同步的要求会影响执行的效率,所以如果不需要线程安全的集合,那么使用 ArrayList 是一个很好的选择,这样可以避免由于同步带来的不必要的性能开销。

可见,如果仅仅是作为数组存储数据,应该选择使用简单数组,使用一个简单的数组(Array)来代替 Vector 或 ArrayList。尤其是对于执行效率要求高的程序更应如此。因为使用数组避免了同步、额外的方法调用和不必要的重新分配空间所带来的消耗。

6.4.4 JSP 内置对象

内置对象是由开发环境所定义的具有特定功能的对象,用户可以直接使用。在 JSP 脚本段中,可以访问这些隐含对象来与 JSP 网页中的可执行 Servlet 环境交互。JSP 中包含了一系列的内置对象,常用的内置对象介绍如下。

1. request 对象

在 JSP 中,内置对象 request 对象是 javax. servlet. HttpServletRequest 类的一个子类对象,当客户端请求一个 JSP 页面时,JSP 容器会将客户端的请求信息及 HTTP 头封装在 request 对象中。request 对象常用的方法见表 6-2。

表 6-2 request 对象常用方法

方 法	说 明
getHeader(String name)	获得 HTTP 定义的文件头的值
getHeaders(String name)	获得一个 HTTP 请求头的所有值
getRemoteAddr()	获取客户端的 IP 地址
getRemoteHost()	获得客户端主机的名字,如果该方法失败,则返回客户端计算机的 IP 地址
getRequestURL()	获得客户端请求的 URL
getRequestURI()	获得客户端请求的 URI
getMethod()	获得请求方法(GET、POST)
getParameter(String name)	返回客户端传送某个请求参数的值,或者是表单数据。name 指定传递的参数名或表单的输入域名
getParameterNames()	获取客户端传来的所有参数的名字,返回值为 Enumeration 类实例
getParameterValues(String name)	获取客户端中参数名为 name 的所有值
getCookies()	返回客户端的 Cookies 对象,结果是一个 Cookie 数组
getQueryString()	返回查询字符串,该字符串由客户端以 GET 方法向服务器端传送。查询字符串出现在页面请求"?"的后面

通过 request 对象可以获取用户访问网页时传入的参数值或表单中的输入。例如,在网页 a. htm 中包含一个表单,其<form>的 action 属性指定表单处理页面为 b. jsp,在 b. jsp 中,可以通过 request. getParameter 方法来获取 a. htm 中的表单输入数据。

要获取一个网页的 userName 参数的值,代码如下:

```
<% String name = request.getParameter("userName"); out.println(name); %>
```

需要说明的是,HTTP 传输默认的编码是 ISO-8859-1,因此在浏览器发出请求时给出的 URL 是编码后的字符串,这样当有中文时,服务器得到的是一个包含乱码的 URL 字符串,在目标页面中要得到正确的中文数据,需要进行代码转换。中文字符编码转换函数如下:

```
<%!
public String codeToString(String str)
{ //中文字符串数据编码转换函数
  String s = str;
  try {
    byte tempB[ ] = s.getBytes("ISO - 8859 - 1");
    s = new String(tempB);
    return s;
  }
  catch(Exception e) {
    return s;
   }
}
%>
```

这样就可以将得到的中文数据转换为正常的中文编码了。例如:

```
<%
String msgtitle = codeToString(request.getParameter("msgtitle"));
String msgcontent = codeToString(request.getParameter("msgcontent"));
%>
```

为了使用方便,可以将上述编码转换函数定义为一个 JavaBean,然后通过在 JSP 页面中导入即可使用。

2. response 对象

在 JSP 中,response 对象是一个 javax. servlet. HttpServletResponse 类的实例,封装了服务器响应客户请求的信息。response 对象的作用是向客户端返回请求,即给客户端传送输出信息,设置表头等。常用的方法见表 6-3。

表 6-3 response 对象常用方法

方 法	说 明
setContentType(String s)	重新设定传回网页的文件格式和编码方式。常用的文件格式有 text/html、text/plain(文本文件)、application/x-msexcel(Excel 文件)和 applicaiton/word(Word 文件)

续表

方　　法	说　　明
addHeader(String name, String value)	添加 HTTP 头,该 Header 将会传到客户端,如果有同名的 Header 存在,原先的 Header 会被覆盖
setHeader(String name, String value)	设定指定名字的 HTTP 头的值,如果该值存在,它将会被新的值覆盖
addCookie(Cookie c)	添加一个 Cookie 对象
setStutus(int n)	设置 HTTP 链接中的服务端响应的状态码
sendError(int sc)	传送状态码
sendError(int sc, String msg)	传送状态码和错误信息
sendRedirect(URL url)	页面重定向,将客户端重新定向到 URL 所指向的页面

使用 response 对象可以设置客户端的页面跳转、页面自动刷新、页面自动跳转等。示例代码如下:

```
<%
    response.setHeader("Refresh","5;URL = http://www.baidu.com");
    response.sendRedirect("a.jsp"); //重定位到 a.jsp 页面
%>
```

上述代码可以使得客户端在 5 秒钟后重新自动跳转到一个新的网址,或直接重新定位到一个新的页面。

需要说明的是,在 sendRedirec()方法中,如果重新定位的网址中含有参数,参数值为中文的话,例如“response.sendRedirect("b.jsp? teachername=张三 &page=1");”在重新定位的页面 b.jsp 中,不能正确获得 teachername 参数的值。

3. Session 对象

所谓会话(Session),是指一个用户和 Web 服务器之间的一次链接。当用户使用浏览器登录到 Web 服务器,并初次浏览一个 JSP 应用的某个网页开始,直到用户离开网站或超时未继续浏览该网站网页为止,之间的浏览操作算作一次会话。

Session 对象是 JSP 为每一个会话而建立的个人对象,可以存储及提供个别用户独享的永久或半永久信息。它是一个与 request 相关的 javax.servlet.http.HttpServletSession 对象。当一个用户首次访问服务器上的一个 JSP 页面时,JSP 引擎产生一个 Session 对象,同时分配一个 String 类型的 ID 号,JSP 引擎同时将这个 ID 号发送到用户端,存放在 Cookie 中,这样 Session 对象和用户之间就建立起一一对应的关系。当用户再次访问连接该服务器的其他页面时,不再为用户分配新的 Session 对象。当用户关闭浏览器时,服务器端保存的该用户的 Session 对象被取消,服务器和用户的对应关系也被取消。如果用户重新打开浏览器再连接到该服务器时,服务器为用户再创建一个新的 Session 对象。

Session 对象常用的方法见表 6-4。

表 6-4 Session 对象常用方法

方法	说明
setAttribute(String name, value)	在 Session 对象中设置属性,并为该属性赋值
getAttribute(String name)	获取 Session 对象指定属性的值,如果该属性不存在,将会返回 null
removeAttribute(String name)	删除指定属性的属性名和属性值
getCreationTime()	返回 Session 对象被创建的时间,单位为毫秒
getId()	返回 Session 对象在服务器端的编号。每生成一个 Session 对象,服务器都会给它一个编号,编号不会重复,服务器根据编号来识别 Session,并且正确地处理某一特定的 Session 及其提供的服务
getLastAccessedTime()	返回当前 Session 对象最后一次被操作的时间,单位为毫秒
getMaxInactiveInterval ()	获取 Session 对象的生存时间,单位为秒
setMaxInactiveInterval (int interval)	设置 Session 对象的有效时间(超时时间),单位为秒。具体值应根据网站的实际应用情况决定,设置几十分钟是很正常的

【例 6-7】 编写一个 JSP 页面,当用户访问站点时,检查是否通过登录页面正常登录,如果正常登录则在 Seesion 对象中添加属性 useraccount,否则,显示一个错误页面,然后强行跳转到站点登录页面。

分析: 根据 Web 的原理,要访问一个页面,只要输入页面对应的 URL 即可,为避免用户任意地访问页面,Web 系统通常要求用户登录,按照系统的提示访问。下面是一个控制页面合法访问的 JSP,文件名为 session-confirm.jsp。

代码清单:

```
<%@ page pageEncoding = "GBK" %>
<%
String useraccount = (String)session.getAttribute("useraccount");
if (useraccount == null) {
  //销毁当前 Session
  session.invalidate();
 %>
<html>
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312">
</head>
<body oncontextmenu = "self.event.returnValue = false" scroll = "no">
<table width = 500 border = '1' align = 'center' cellspacing = '2' bgcolor = 'E0E0E0'>
<tr height = 30 >
   <td colspan = "2" align = "right" bgcolor = '#0066CC'>
      <b><a href = "javascript:history.go( - 1)"><font color = "000000">×</font></a></b>
   </td>
</tr>
<tr height = 30>
   <td width = "100" rowspan = "3" align = "center" >
       <font face = 'Wingdings' color = '#FF0000' style = 'font - size:32pt'>L</font>
   </td>
   <td>造成错误的原因可能是</td>
</tr>
<tr height = 30>
     <td>未进行正常登录,请重新登录</td>
```

```
</tr>
<tr height = 30>
    <td>系统连接超时,请重新登录</td>
</tr>
<tr height = 30>
    <td colspan = "2" align = "center"><a href = 'javascript:history.go( - 1)'>返 回</a></td>
</tr>
</table>
</body>
</html>
<%
//重定向到 Web 应用的首页
response.sendRedirect(request.getContextPath() + "/index.jsp");
}
 %>
```

在其他网页中,可以包含该页面,这样就可以保证页面不能被直接访问了。

4. application 对象

application 对象可存储并提供给一组 JSP 应用所有用户的共享信息,有效范围为构成该 JSP 应用的所有 JSP 页面,可以实现不同页面之间的数据共享。一般情况下,可以将 application 对象作为一个存储许多共用对象的容器,这是一个 javax. servlet. ServletContext 对象。

application 对象常用的方法见表 6-5。

表 6-5　application 对象常用方法

方　法	说　明
void setAttribute(String key, Object obj)	将参数 Object 指定的对象 obj 添加到 application 对象中,并为添加的对象指定了一个索引关键字 key,如果添加的两个对象的关键字相同,则先前添加的对象被清除
Object getAttribute(String key)	获取 application 对象中含有关键字是 key 的对象。由于任何对象都可以添加到 application 对象中,因此用该方法取回对象时,应强制转化为原来的类型
removeAttribute(String key)	从当前 application 对象中删除关键字是 key 的对象
String getservletInfo()	获取 servlet 编译器的当前版本的信息

使用 application 对象,可以让多个 JSP、servlet 共享数据。例如,有两个 JSP 页面 a. jsp 和 b. jsp。在页面 a. jsp 中包含如下代码:

```
<%@ page contentType = "text/html;charset = GBK" %>
<%
    String str = "你好";
    application.setAttribute("greeting",str);
 %>
```

在另一个页面 b. jsp 中可以访问 application 对象的属性。代码如下:

```
<%@ page contentType = "text/html;charset = GBK" %>
```

```
<%
    String str;
    str = (String)application.getAttribute("greeting");
    out.print(str);
%>
```

此外，使用 application 对象，还可以获得 Web 应用配置参数。在 JSP 页面中，访问数据库所使用的驱动、URL、用户名、密码可以写在配置文件 web.xml 中。例如，下面是一个 web.xml 数据：

```
<context-param>
    <param-name>driver</param-name>
    <param-value>com.mysql.jdbc.Driver</param-value>
</context-param>
<context-param>
    <param-name>url</param-name>
    <param-value>jdbc:mysql://localhost:3306/javaee</param-value>
</context-param>
<context-param>
    <param-name>user</param-name>
    <param-value>root</param-value>
</context-param>
<context-param>
    <param-name>pass</param-name>
    <param-value>root</param-value>
</context-param>
```

通过这种方式，将配置信息放在 web.xml 文件中进行配置，避免使用程序编码方式写在代码中，可以更好地提高程序的移植性。在操作数据库时，通过 application 对象，可以获得这些参数的配置情况。

【例 6-8】 编写 JSP 页面，获取 Web 系统的数据库具体配置信息，并输出一个数据库表。

分析：数据库的配置信息可以通过 Web 站点的 web.xml 进行配置，该配置信息通过 JSP 的内置对象 application 获取。

代码清单：

```
<%@ page language="java" import="java.util.*" pageEncoding="gb2312"%>
<%@ page import="java.sql.*"%>
<%
//从配置参数中获取驱动
String driver = application.getInitParameter("driver");
//获取数据库 URL
String url = application.getInitParameter("url");
//获取用户名和密码
String user = application.getInitParameter("user");
String pass = application.getInitParameter("pass");
//注册驱动
Class.forName(driver);
//获取数据库连接
```

```
Connection conn = DriverManager.getConnection(url,user,pass);
//创建 Statement 对象
Statement stmt = conn.createStatement();
//执行查询
ResultSet rs = stmt.executeQuery("SELECT * FROM newsinfo");
%>
<table bgcolor = "9999dd" border = "1" align = "center">
<%
//遍历结果集
while(rs.next()) {
%>
    <tr>
        <td><% = rs.getString(1) %></td>
        <td><% = rs.getString(2) %></td>
    </tr>
<%
}
%>
</table>
```

对于 application 对象，当站点服务器开启的时候，application 就被创建，直到网站关闭。因此，application 对象可能持续地存在几个月甚至更长的时间，可以用于实现站点访问记数器等功能。例如，在站点首页中增加如下代码：

```
<%
  Integer number = (Integer)application.getAttribute("Count");
  if (number == null) {
      number = new Integer(1);
      application.setAttribute("Count",number);
  }
  else{
      number = new Integer(number.intValue() + 1);
      application.setAttribute("Count",number);
  }
  %>
您是第<% = (Integer)application.getAttribute("Count") %>位访问者
```

5. out 对象

out 对象用于发送输出流，作用是将结果输出到客户端。它最常用的方法有两个：print()和 println()。输出换行符使用 newLine()方法。

例如，不用表达式，可以直接访问隐含对象 out 来输出信息：

```
<% out.println("<h1 align = 'center'> Hello</h1>"); %>
```

通常情况下，使用 out.print()可以输出 html 代码，以动态地构造 html 页面内容，如生成各种表格等。

6. 其他对象

在 JSP 中，内置对象还有：①config 对象，用于传递在 Servlet 程序初始化时所需的信

息；②pageContext 对象，该对象提供了对页面上的所有对象以及命名空间的访问，用于管理网页属性；③page 对象，当前页面，相当于 Java 中的 this；④exception 对象，错误处理对象，用于处理捕捉到的异常。

【例 6-9】 编写一个 JSP 文档，完成一个登录界面，输入用户名和密码，如果输入 guest 则转移到一个默认的首页，如果不输入用户名，则重新回到该页，直到输入正确的用户名和密码，此时重定向到注册用户网页。

分析：在一个 JSP 中，包含表单，表单的 action 属性可以指定执行本页本身，这样可以实现一些特殊的效果。

代码清单：login.jsp

```
<%@ page contentType = "text/html;charset = gb2312" %>
<html>
<body>
<%
String userName = request.getParameter("userName");
if (userName! = null && !"".equals(userName))
    response.sendRedirect("http://gsl.sdu.edu.cn/");
%>
<form name = "form1" action = "login.jsp" method = "post">
<table border = "2" width = "300" cellspacing = "0" cellpadding = "0"  align = "center">
<tr height = "50">
    <td align = "center">用户登录</td>
</tr>
<tr height = "35">
   <td style = "font - size:13">用户名: <input type = "text" name = "userName"></td>
</tr>
<tr height = "35">
   <td style = "font - size:13">密   码: <input type = "text" name = "userPassword">
</td>
</tr>
<tr height = "40">
   <td align = center><input type = "submit" value = "登录"></td>
</tr>
</table>
</form>
</body>
</html>
```

将文件保存到 d:\haobook 文件夹中，在浏览器地址栏中输入该页面的网址：http://127.0.0.1/book/login.jsp，即可调用显示。页面的执行情况分析如下。

第一次执行该文档时，由于 URL 中没有设置 userName，并且表单尚未生成，因此在开始处的 JSP 段，执行 userName = request.getParameter("userName")时，参数 userName 不存在，此时会返回 null(空和字符串"null"不同)，此时，后面的页面重定位没有执行，接着会显示后面的 html 代码，显示登录表单。

当登录表单显示后，如果此时没有输入用户名，直接单击"登录"按钮，表单提交，执行 action 设置中设置的页面 login.jsp，即本页面本身。此时表单已经存在，即 userName 输入

域已经存在，此时在开始处的 JSP 代码段，读取的 userName 不再是 null，而是空字符("")，即 userName 此时为空字符，即"". equals(userName)条件为真。

【例 6-10】 编写 JSP 代码，显示站点的在线人数。

分析：显示在线人数的方法很多，前面我们看到用 application 对象可以显示站点的访问人数，但是系统重启后，数据会丢失。对于在线人数，利用 Session 对象的数量可以获取准确的在线人数。因此，可以编写一个类来对 Session 的创建和注销进行记录，从而得到在线人数的数据。

首先编写一个统计会话人数的 java 类，代码清单如下(文档名 SessionCounter. java)：

```
package pub;
import javax.servlet.*;
import javax.servlet.http.*;
public class SessionCounter implements HttpSessionListener
{
  private static int activeSessions = 0;
  public void sessionCreated(HttpSessionEvent se)
  {
    activeSessions++;
  }
  public void sessionDestroyed(HttpSessionEvent se)
  {
    if (activeSessions > 0)
       activeSessions--;
  }
  public static int getActiveSessions()
  {
    return activeSessions;
  }
}
```

将上述 Java 类文件存储在 Web 应用的用户自定义类文件夹 classes 中的 pub 包(即一个子文件夹 pub)中，即存储到 d:\haobook\WEB-INF\classes\pub 中，然后编译该文件。

编写调用该 SessionCounter. java 的 JSP 文档，文档名为 myonline. jsp，内容如下：

```
<%@ page import="pub.SessionCounter" %>
<html>
<head>
  <meta http-equiv="refresh" content="60">
</head>
<body>
在线人数: <%=SessionCounter.getActiveSessions() %>
</body>
</html>
```

最后，修改 d:\haobook\WEB-INF\web. xml 配置文件，在<web-app>…</web-app>元素内，添加如下内容：

```
<!-- Listeners -->
<listener>
```

```
        <listener-class>
            pub.SessionCounter
        </listener-class>
</listener>
```

6.4.5 在 JSP 中使用 JavaBean

JavaBean 主要应用于 JSP 网页中,通过 JavaBean 可以更好地将业务逻辑代码和 JSP 的 HTML 标记进行分离,便于系统的维护。在 JSP 中使用 JavaBean,通常有两种方法。

1. 使用 <jsp:useBean> 标记符访问 JavaBean

在 JSP 中,使用 <jsp:useBean> 标记符访问 JavaBean 的一般形式是:

```
<jsp:useBean id="实例名" class="包.类" scope="page|request|session| application" />
```

其中,id 指定一个 JavaBean 类的实例名,如果这个实例已经存在,将直接引用这个实例;如果这个实例不存在,将通过后面 class 参数中的定义新建一个类的实例。class 参数设置存储 JavaBean 的路径和类的名称,例如 class="cards. NameCard",则表明要使用 Web 应用根目录中"WEB-INF\classes\cards"下的一个 NameCard. class 文件,其中 cards 是包名,NameCard 为 JavaBean 对应的 Java 类名。scope 用于定义实例存在的范围,事实上这定义了这个实例所绑定的区域及其有效范围。

(1) page,这个 JavaBean 将存在于该 JSP 文件以及此文件中的所有静态包含文件中,直到页面执行完毕为止。

(2) request,这个 JavaBean 将作为一个对象绑定于该页面的 request 中,即该 JavaBean 在该页面发出的请求中有效。

(3) session,这个 JavaBean 将作为一个对象绑定于 session 中,即该 JavaBean 在本地有效。

(4) application,这个 JavaBean 将作为一个对象绑定于 application 中,即该 JavaBean 在本应用中有效。

2. 嵌入 java 代码方式访问 JavaBean

JavaBean 是一个 Java 类,在 JSP 中也可以使用导入(import)Java 类的方法使用 JavaBean。过程如下。

(1) 首行导入 JavaBean。例如要导入的 JavaBean 为 XXX,语句如下:

```
<%@ page import="com.javaBean.XXX" %>
```

(2) 下面就可以像在 Java 语言中一样来使用这个 JavaBean 了。例如:

```
<% XXX obj = new XXX(); %>
```

【例 6-11】 编写一个 JavaBean,来读取当前系统的日期和时间,并返回相应的字符串。

分析: 我们编写一个关于时间处理的 JavaBean,关于时间操作是 JSP 中经常使用的功能。

代码清单：mytime.java

```
package pub;
public class mytime
{
   public String getDatetime()
   {
     String datestr = "" ;
     try {
       java.text.DateFormat df = new java.text.SimpleDateFormat("yyyy-MM-dd HH:mm") ;
       java.util.Date mydate = new java.util.Date() ;
       datestr = df.format(mydate) ;
     }
     catch (Exception ex) {
     }
     return datestr ;
   }
   public String getTime()
   {
       String timestr = "";
       try {
          java.text.DateFormat tf = new java.text.SimpleDateFormat("HH:mm:ss");
          java.util.Date ud = new java.util.Date();
          timestr = tf.format(ud);
       }
       catch(Exception ex) {
       }
     return timestr;
   }
}
```

从上述例子中的 mytime 的定义可以看出，JavaBean 的定义并没有那么严格，可以没有属性，属性也不一定都要有 set 和 get 方法。

在 Web 应用中，用户定义的类或 JavaBean 通常存储在根目录下的“WEB-INF\classes\包\”文件夹中，Java 通过包来实现用户类定义的分类存储和管理。

上述 JavaBean 应存储在 Web 应用特定的文件夹中，即 Web 应用根目录中“WEB-INF\classes\pub”下，用 javac mytime.java 编译生成 mytime.class，该文件将存储在 pub 包中，即 pub 子文件夹中。

【例 6-12】　编写一个 JSP 页面，使用例 6-11 中定义的 JavaBean 以及例 6-4 中定义的 JavaBean，即 NameCards。

分析：JavaBean 主要应用在 JSP 页面中，通常通过<jsp:useBean>标记来应用。调用 JavaBean 很简单。下面的代码演示了例 6-11 和例 6-4 中定义的两个 JavaBean 在一个 JSP 文档中的使用。

代码清单：mybeans.jsp

```
<%@ page contentType = "text/html;charset = GBK" %>
<%@ page language = "java" %>
<jsp:useBean id = " mycard " class = "cards.NameCard" scope = "application" />
```

```
<jsp:useBean id="mytime" class="pub.mytime" scope="application" />
<html>
<body>
<%
  mycard.setName("Hao XW");
  mycard.setAddress("No.27 Shanda South Road,Jinan,China");
 %>
姓名:<% =mycard.getName() %><br>
地址:<% =mycard.getAddress() %><br>
日期:<% =mytime.getDatetime().substring(0,16) %><br>
</body>
</html>
```

在上述的 JSP 页面中,调用了一个 JavaBean,即 cards.NameCard。将上述 JSP 文档保存到 Web 应用的主目录下,然后在浏览器地址栏中输入 http://127.0.0.1/book/mycard.jsp,则浏览器将显示运行结果。

6.5 JDBC 与数据库编程

在计算机应用软件中,数据的存储可分为文件和数据库两种模式。一般的应用软件,例如 Word、Excel、Photoshop、Flash 等,都有注册的文档类型,采用特定格式的文件存储数据。在 Web 应用中,通常使用数据库来存储和管理系统数据。为了实现应用系统和数据库管理系统的相对独立,应用系统通过相应的数据接口来访问和操作数据库。

6.5.1 JDBC 接口及其安装和配置

Java 中连接数据库的技术是 JDBC(Java DataBase Connectivity),通过 java.sql 包中的类、接口和异常事件,可以对数据库进行操作。在 JDK 中,java.sql 包定义了访问数据库的接口和类,但是 JDBC API 不能直接访问数据库,必须依赖于数据库厂商提供的针对其具体的数据库产品的 JDBC 驱动程序。大多数的数据库管理系统都带有和 Java 相配的 JDBC 驱动程序,Java 程序通过 JDBC 驱动程序即可实现与数据库的连接,执行查询、提取数据等操作,使得 Java 程序能访问诸如 Oracle、MS SQL Server、MySQL 和 MS Access 等数据库。

1. JDBC 驱动的安装和配置

要使用 JDBC 驱动,首先要在系统中进行安装和配置。在 Web 应用中,WEB-INF\文件夹中包括 classes 子文件夹和 lib 子文件夹,其中 classes 文件夹存储用户定义的类,lib 文件夹存储应用系统所需要的驱动程序包.jar 文件。

在 WEB-INF\classes 文件夹下,可以进一步再定义一些子文件夹(即用户包),用于存储用户定义的 Java 类和 JavaBean 源文件和.class 文件,例如,定义 WEB-INF\classes\pub 文件夹下,存储关于数据库操作的 JavaBean。

对于数据库的 JDBC 驱动应复制到 WEB-INF\lib 文件夹中,驱动程序可以从网上查找并下载,例如:MS SQL Server 2005 的驱动为 sqljdbc.jar。最后,在系统环境变量的 classpath 中添加驱动所在的目录,在 classpath 设置的后面,添加下面的路径:用户系统所

在的驱动器:根目录\WEB-INF\lib\,例如:d:\haobook\WEB-INF\lib\。

2. 数据库访问相关的类

JDBC API 主要位于 java. sql 包中,java. sql 包定义了所有的 JDBC 应用相关的类和方法。常用的类和接口如下。

(1) java. sql. DriverManager 类:主要用于处理 JDBC 驱动程序的调入,并且对新的数据库连接提供支持。常用方法如下。

① getConnection(String url,String user,String pwd),建立和数据库服务器程序的连接,返回一个 Connection 类对象。

其中,url 是指要连接的数据库名,例如:"jdbc:sqlserver://localhost:1433;DatabaseName= gsl_messageboard",user 和 pwd 对应该数据库服务器的用户名和密码。可以通过数据库服务器的管理界面得到。

② setLoginTimeOut(int seconds),设定等待数据库服务器连接的最长时间。

③ setLogWriter(PrintWriter out),设定输入数据库日志的 PrintWriter 对象。

(2) java. sql. Connection 类:代表和数据库的连接,用户通过该对象操作特定的数据库。常用方法如下。

① getMetaData(),返回数据库的 MetaData 数据。MetaData 数据包含了数据库的相关信息,例如:当前数据库连接的用户名、使用的 JDBC 驱动程序、数据库允许的最大连接数、数据库的版本等。

② createStatement(),创建并返回一个 Statement 类对象。

③ PrepareStatement(String sql),创建并返回 prepareStatement 对象。

在实际应用中,我们会遇到同时提交多个 SQL 语句,这些 SQL 语句要么全部成功,要么全部失败,如果其中一条提交失败,则必须撤销整个事务。为此,Connection 类还提供了 3 个控制事务的方法。

① setAutoCommit(boolen autoCommit),设置是否自动提交事务,默认为自动提交。

② commit(),提交事务。

③ rollback(),撤销事务。

(3) java. sql. Statement 类:用来执行静态 SQL 语句。例如,调用 executeQuery(String sql)方法可以执行 select 语句,并返回一个 ResultSet 类对象,保存了查询结果数据集。调用 executeUpdate(String sql)方法可以执行 insert、update、delete 语句,即进行数据记录的插入、修改和删除操作。

(4) java. sql. PrepareStatement 类:用于执行动态的 SQL 语句,即允许 SQL 语句中包含参数。使用方法为:

```
String sql = "SELECT col1 FROM tablename WHERE col2 = ? AND col3 = ? ";
PrepareStatement perpStmt = conn.preparestatement(sql);
perpStmt.setString(1,col2Value);
perpStmt.setFloat(2,col3Value);
ResultSet rs = perpStmt.executeQuery();
```

(5) java. sql. ResultSet 类:用来保存 select 语句查询得到的记录集,用它可以浏览和

存取数据库内的记录。一个 Statement 对象在同一时刻只能打开一个 ResultSet 对象。

通过 ResultSet 的 getXXX()方法来得到字段值,ResultSet 提供了 getString()、getFloat()、getInt()、getTime()等方法。可以通过字段的序号或者字段的名字来指定要获取的某个字段的值。在上例中,使用 getString(0)、getString("col1")都可以获得字段 col1 的字符串值,通过 Integer 对象可以将字符串值转换为整数。

有些字段可以使用不同的方法返回值,但返回值的类型可以不同,例如对于一个 int 字段,可以使用 getString()和 getInteger()来获取字段的值,但前者返回一个字符串,后者则返回一个整数值。再如:对于一个 Datetime 型字段,可以使用 getString()、getDate()、getTime()返回数据,但返回数据的类型不同。因此,在选择 get 方法提取数据字段值时,应根据程序的需要选择相应的方法,可以节省数据类型转换。

6.5.2 结构化查询语言基础

结构化查询语言(Structured Query Language,SQL)是关系数据库管理系统的标准语言,用于存取数据以及查询、更新和管理关系数据库系统。不同的关系数据库使用的 SQL 版本有一些差异,但大多数都遵循 ANSI SQL 标准。

1. SQL 的组成与 T-SQL

SQL 语言包含三部分。

(1) 数据定义语言(Data Definition Language,DDL):用于定义关系数据库的模式、外模式和内模式,以实现对数据库基本表、视图及索引文件的定义、修改和删除等操作。常用的 DDL 语句是 CREATE、DROP 和 ALTER 命令。

(2) 数据操作语言(Data Manipulation Language,DML):用于完成数据查询和数据更新操作。其中数据更新指对数据进行插入、删除和修改操作。常用的 DML 语句是 SELECT、INSERT、UPDATE 和 DELETE 命令。

(3) 数据控制语言(Data Control Language,DCL):用于控制对数据库的访问,服务器的关闭、启动等操作。常用的 DCL 语句有 GRANT、REVOKE、COMMIT、ROLLBACK 等命令。

在实际应用中,使用较多的是 Transact-SQL(T-SQL),它是微软遵循 ANSI SQL-92 标准在 Microsoft SQL Server 系统中使用的语言。Transact-SQL 分成五部分,即数据定义语言、数据操纵语言、数据控制语言、事务管理语言和附加的语言元素。其中,附加的语言元素主要包括标识符、变量和常量、运算符、表达式、数据类型、函数、控制流语言、错误处理语言、注释等。在 T-SQL 中,定义了大量的标准函数,涉及聚合函数、数学函数、字符串函数、日期和时间函数、系统函数、游标函数和元数据函数,这就给了 SQL 以强大的编程能力。关于这些函数的定义及功能,超出本书的写作范围,请参考专门的 T-SQL 书籍。

2. 简单数据查询

查询语句用来对已经存在于数据库中的数据按照特定的组合、条件表达式或者一定次序进行检索。SELECT 语句的完整语法较复杂,其主要子句可归纳如下:

```
SELECT <select_list>
[INTO <new_table> ]
FROM <table_source>
[WHERE <search_condition> ]
[GROUP BY <group_by_expression> ]
[HAVING <search_condition> ]
[ORDER BY <orderby_expression> [ASC | DESC]]
[COMPUTE { {AVG | COUNT | MAX | MIN | SUM } (expression) } [ BY expression]]
```

在 SQL 中，Select 语句非常复杂，但常用的子句主要是 SELECT、FROM 和 WHERE 子句。SELECT 子句用于指定将要查询的列名称，FROM 子句指定这些数据来自哪些表或视图，WHERE 子句则用于指定数据应该满足的条件。在一般的检索中，SELECT 子句和 FROM 子句是必不可少的。

1) SELECT 子句

SELECT 子句的功能是指定要查询的结果列。其语法格式如下：

```
SELECT [ ALL | DISTINCT ] [ TOP <n> [ PERCENT ] ] <select_list>
```

其中 select_list 定义为：

```
<select_list> ::= { *
                      | {table_name | view_name | table_alias }. *
                      | {column_name | expression } [ [ AS ] column_alias ]
                      | column_alias = expression
                  }
```

各参数说明如下。

(1) ALL：返回结果集中的所有行，是系统默认值。

(2) DISTINCT：指明结果集中如果有值相同的行，则只显示其中的一行；对于 DISTINCT 关键字来说，Null 值是相等的。

(3) TOP<*n*> [PERCENT]：指明仅返回结果集中的前 *n* 行，如果有[PERCENT]，则返回结果集中的百分之 *n* 行记录。

对于 select_list，指定要查询的结果列，结果列表是以逗号分隔的一系列表达式。

(4) *：表示返回在 FROM 子句中包括的所有对象的全部列，这些列按 FROM 子句中指定的表或视图顺序返回，并对应于它们在表或视图中的顺序。

(5) { table_name | view_name | table_alias }. *：指明返回指定表或视图的全部列。

(6) column_name：指明返回的列名。

(7) expression：是一个由列名、常量、函数通过操作符连接起来的数据表达式，作为返回结果集中的一列。

(8) column_alias：指明用以代替出现在结果集中的列名或表达式的别名。

【例 6-13】 假定有一个数据表 student，存储学生的简单信息，包含的字段有 sno、sname、ssex、sdept、sbirthday，对数据进行基本查询。

分析：本题在于练习 SELECT 子句的不同应用，进一步理解各个参数的含义和功能。

查询 1：查询所有学生的详细信息。查询语句为：

```
SELECT sno,sname,ssex,sdept, sbirthday FROM student
```

或

```
SELECT * FROM student
```

查询 2：查询有过选课记录的学生的学号，消除取值重复的行。查询语句为：

```
SELECT DISTINCT sno FROM sc
```

查询 3：查询所有学生的编号、姓名和所在系，并修改结果集中列的名称。语句如下：

```
SELECT '学号' = sno, '姓名' = sname,sdept AS '院系' FROM student
```

2）WHERE 子句与条件查询

查询满足指定条件的记录可以通过 WHERE 子句实现。WHERE 子句指定数据检索的条件，以限制返回的数据行。其语法格式如下：

```
WHERE <search_condition>
```

search_condition 用于指定查询条件。WHERE 子句中常用的查询条件如表 6-6 所示。

表 6-6 常用的查询条件

查询条件	谓词
比较	=、>、<、>=、<=、!=、<>、!>、!<
确定范围	BETWEEN AND、NOT BETWEEN AND
确定集合	IN、NOT IN
字符匹配	LIKE、NOT LIKE
空值	IS NULL、IS NOT NULL
多重条件	AND、OR、NOT

使用谓词 LIKE 进行字符串的匹配，其一般的语法格式如下：

```
[ NOT ] LIKE '<匹配串>'
```

其含义是查找指定的属性列值与<匹配串>相匹配的记录。<匹配串>可以是一个完整的字符串，也可以含有通配符。在 SQL Server 中，有如表 6-7 所示的四种通配符。

表 6-7 通配符

通配符	功能	示例
%	代表零或多个字符	"sn%"，表示"sn"后可接任意字符串
-	代表一个任意字符	"s_n"，表示"s"与"n"之间可以有一个字符
[]	表示在某一范围内的字符	[1-9]，表示 1～9 之间的字符
[^]	表示不在某一范围内的字符	[^1-9]，表示不在 1～9 之间的字符

需要注意的是，如果使用 WHERE 子句来限制查询的范围，WHERE 子句必须紧跟在 FROM 子句之后。下面是一组简单的条件查询例子。

【例 6-14】 使用条件查询，对上述例子中的数据表进行查询。

查询 1：查询性别为“女”的学生的学号、姓名和系别。

```
SELECT sno,sname,sdept
FROM student
WHERE ssex = N'女'
```

查询 2：查询成绩在 80～100 之间的学生及课程信息。

```
SELECT * FROM sc
WHERE grade BETWEEN 80 AND 100
```

查询 3：查询出生日期不在 1990—1992 年之间的学生的姓名、性别和所属系。

```
SELECT sname,ssex,sdept
FROM student
WHERE sbirthday NOT BETWEEN '1990 - 01 - 01' AND '1992 - 01 - 01'
```

查询 4：查询信管系、会计系、营销系学生的姓名和性别。

```
SELECT Sname,Ssex
FROM   Student
WHERE Sdept IN ( N'信管系',N'会计系',N'营销系');
```

查询 5：查询姓“张”的学生姓名。

```
SELECT sname
FROM student
WHERE sname LIKE N'张 % '
```

查询 6：查询信管系的女生的学号、姓名。

```
SELECT sno,sname
FROM student
WHERE sdept = N'信管系' AND ssex = N'女'
```

3）ORDER BY 子句与排序输出

用 ORDER BY 子句对查询结果按照一个或多个属性列的升序(ASC)或降序(DESC)排列,空值被视为最低的可能值。

例如：查询所有男生的学号、姓名和出生日期,并按照出生日期降序排列；若出生日期相同,则按学号升序排列。则 SQL 语句为：

```
SELECT sno,sname,sbirthday
FROM student
WHERE ssex = N'男'
ORDER BY sbirthday DESC,sno
```

3．聚合查询

聚合查询是指通过查询对一组数据进行聚合运算得到聚合值的过程。在聚合运算中主要是使用聚合函数。SQL Server 中提供了许多聚合函数,常用的聚合函数见表 6-8。

在 Microsoft SQL Server 2005 中,可以在三个子句中使用聚合函数,即 SELECT 子句、COMPUTE 子句和 HAVING 子句,在此先介绍如何在 SELECT 子句和 COMPUTE 子句中使用聚合函数。

表 6-8 常用的聚合函数

函 数	功 能
COUNT([DISTINCT | ALL] *)	统计记录个数
COUNT([DISTINCT | ALL]<列名>)	统计一列中值的个数
SUM([DISTINCT | ALL]<列名>)	计算一列值的总和(此列必须是数值型)
AVG([DISTINCT | ALL]<列名>)	计算一列值的平均值(此列必须是数值型)
MAX([DISTINCT | ALL]<列名>)	求一列值中的最大值
MIN([DISTINCT | ALL]<列名>)	求一列值中的最小值

1) SELECT 子句中的聚合

在 SELECT 子句中可以使用聚合函数进行运算,运算结果作为新列出现在结果集中。在聚合运算的表达式中可以包括列名、常量以及由算术运算符连接起来的函数。下面是一组使用聚合查询的示例。

查询 1:统计所有的学生人数。

```
SELECT COUNT( * ) AS 学生人数
FROM student
```

查询 2:统计有选课记录的学生人数。

```
SELECT COUNT(DISTINCT sno)  AS 有选课记录的学生人数
FROM sc
```

查询 3:统计选修“c01”课程的平均成绩、最高成绩和最低成绩。

```
SELECT AVG(grade) AS c01 课程的平均成绩,
       MAX(grade) AS c01 课程的最高成绩,
       MIN(grade) AS c01 课程的最低成绩
FROM sc
WHERE cno = 'c01'
```

2) COMPUTE 子句中的聚合

在 SELECT 子句中出现聚合函数时,结果集中的数据全是聚合值,没有明细值。COMPUTE 子句不仅可以使用聚合函数计算聚合值,而且可以依然保持原有的明细值,新的聚合值作为附加的汇总列出现在结果集的最后。需要注意的是,如果是用 COMPUTE 子句指定的行聚合函数,则不允许它们使用 DISTINCT 关键字。

例如:查询信管系学生的学号、姓名和年龄,最后汇总出该系学生的总人数和平均年龄。

```
SELECT sno,sname,year(getdate( )) - year(sbirthday) as 年龄
FROM student
WHERE sdept = N'信管系'
COMPUTE COUNT(sno),AVG(year(getdate( )) - year(sbirthday))
```

4. 分组查询

GROUP BY 子句对查询结果按照一定条件进行分组,分组子句通常与 SQL Server 提

供的聚合函数一起使用。对查询结果分组的目的是细化聚合函数的作用对象，如果未对查询结果分组，则聚合函数将作用于整个查询结果，分组后聚合函数将作用于每一个组，每一个组都有一个函数值。SELECT 语句后的输出列只能是聚合函数和分组列。

如果分组后还需要按一定的条件对这些组进行筛选，最终只输出满足指定条件的组，则可以使用 HAVING 子句指定筛选条件。

例如：统计所有被选过的课程的课程号及被选过的次数。

```
SELECT cno,count( * ) AS 被选次数
FROM sc
GROUP BY cno
```

如果要统计被选过 2 次和 2 次以上的课程的课程号及被选过的次数，则需要使用 HAVING 子句。代码如下：

```
SELECT cno,count( * ) AS 被选次数
FROM sc
GROUP BY cno
HAVING COUNT( * )> = 2
```

5. 连接查询

前面的查询都是针对一个表进行的，若一个查询同时涉及两个或两个以上的表，并且每一个表中的数据往往作为一个单独的列出现在结果集中，则称之为连接查询。

FROM 子句指定需要进行数据查询的表，其语法格式如下：

```
FROM < table_source >
```

其中 table_source 定义为：

```
< table_source >: : = table_name[ [ AS]table_alias] | view_name [ [AS] view_alias]
                    | derived_table [ [AS] table_alias]
                    | < joined_table >
```

各参数说明如下。

(1) table_source：指明 SELECT 语句所使用的表、视图等数据源。

(2) table_name [[AS] table_alias]：指明表名和表的别名。

(3) view_name [[AS] view_alias]：指明视图名和视图的别名。

(4) derived_table [[AS] table_alias]：是从指定的数据库和表中检索的子查询结果。

(5) joined_table：指明由连接查询生成的查询结果作数据源。对于多个连接，可使用圆括号来更改连接的自然顺序。

根据多个表返回结果集的处理不同，连接查询又分为以下几种情况。

(1) 交叉连接。交叉连接不使用任何连接条件来限制结果集合，而是对两个数据源中的行以所有可能的方式进行组合，也就是做广义笛卡儿积。交叉连接在查询结果集中，包含了所连接的两个表中所有行的全部组合，其结果的行数等于两个表行数之积，列数等于两个表列数之和。

例如,对 student 表和 course 表做交叉连接,可以书写下面的 SQL 语句:

```
SELECT student. * ,course. * FROM student,course
```

也可以用 JOIN 关键字表示交叉连接,将上述语句写为:

```
SELECT student. * ,course. * FROM student CROSS JOIN course
```

(2) 内连接。内连接是用比较运算符比较表中列值,返回符合连接条件的数据行,从而将两个表连接成一个新表。内连接根据不同的情况可分为以下几种。①等值连接:在连接条件中使用等号,通过相等的字段连接起来的查询称为等值连接。②自然连接:若在等值连接中,把目标列中重复的字段去掉则称为自然连接。它是一种特殊的等值连接。③非等值连接:表之间的连接,如果使用除了"="之外的连接符连接起来的查询称为非等值连接。虽然 SQL Server 提供了非等值连接查询,但非等值连接查询的例子很少有实际应用价值。④自连接:连接可以是一个表与其自身进行的,这样的连接称为自连接。

【例 6-15】 假设有一个学生选课数据表 sc,包含 sno、cno 和 grade 三个字段。现在要求查询工商系有选课记录的学生的全部信息以及他们的选课信息。

分析:查询涉及两个数据表,需要使用连接查询。

代码清单:

```
SELECT student. * ,sc. *
FROM student,sc
WHERE student.sdept = N'工商系' AND student.sno = sc.sno
```

查询结果示例如下:

	sno	sname	ssex	sdept	sbirthday	sno	cno	grade
1	2008002	王小惠	女	工商系	1991-03-12 00:00:00.000	2008002	c03	90.0
2	2008002	王小惠	女	工商系	1991-03-12 00:00:00.000	2008002	c04	89.0
3	2008005	张小均	女	工商系	1993-07-06 00:00:00.000	2008005	c01	95.0

【例 6-16】 如果再增加一个课程数据表 course,包含 cno、cname 字段,查询信管系选修"C 语言"课程的学生的姓名、课程名和成绩。

分析:现在是三个数据表的查询,可以用谓词表示等值连接,也可以用 JOIN 关键字表示等值连接。SQL 语句如下。

① 用谓词表示等值连接:

```
SELECT sname, cname, grade
FROM student S, course C, sc
WHERE S.sno = sc.sno AND C.cno = sc.cno
AND sdept = N'信管系'AND cname = N'C 语言'
```

② 用 JOIN 关键字表示等值连接:

```
SELECT Sname, Cname, Grade
FROM Student S JOIN SC JOIN Course C ON SC.Cno = C.Cno ON S.Sno = SC.Sno
WHERE sdept = N'信管系'AND cname = N'C 语言'
```

(3) 外连接。前面内连接所举的例子中,连接的结果是从多个表的组合中筛选出符合连接条件的数据,如果数据无法满足连接条件,则将其丢弃。但是外连接则不然,在外连接

中，不仅包括那些满足条件的数据，而且某些不满足条件的数据也会显示在结果集中。也就是说，外连接只限制其中一个表的数据行，而不限制另外一个表中的数据。这种连接形式在许多情况下是非常有用的，例如在连锁超市统计报表时，不仅要统计那些有销售量的超市和商品，还要统计那些没有销售量的超市和商品。

需要注意的是，外连接只能用于两个表中。两个表有主次之分，以主表的每行数据去匹配从表的数据列，符合连接条件的数据将直接返回到结果集中，对那些不符合连接条件的列将被赋予 NULL 值后再返回到结果集中。根据两个表的主次关系，外连接分为左外连接、右外连接、完全外连接。

6. 嵌套查询

一个 SELECT-FROM-WHERE 语句称为一个查询块，有时一个查询块无法完成查询任务，需要一个子查询块的结果作为父查询块的条件。将一个查询块嵌套在另一个查询块的条件子句中的查询称为嵌套查询。嵌套查询使我们可以用多个简单查询构成复杂的查询，从而增强查询功能。

例如，查询选过“c01”号课程的学生的学号、姓名和所在系：

```
SELECT sno,sname,sdept
FROM student
WHERE sno IN (SELECT sno FROM sc WHERE cno = 'c01')
```

7. 组合查询

查询语句的结果集往往是一个包含了多行数据的集合。集合之间可以进行并、交、差等运算。其中，UNION 运算符表示并集运算，EXCEPT 运算符表示差运算，INTERSECT 运算符表示交运算。需要注意的是，在集合运算时，所有查询语句中列的数量和顺序必须相同，且数据类型必须兼容。

例如，查询所有的男生和信管系的学生的姓名、性别和所在系，使用 UNION 操作符，查询语句如下：

```
SELECT sname,ssex,sdept FROM student WHERE ssex = N'男'
UNION
SELECT sname,ssex,sdept FROM student WHERE sdept = N'信管系'
```

8. 数据的插入

在 Transact-SQL 中，数据插入语句 INSERT 有两种形式：一种是插入一条记录，另一种是插入子查询结果。向表中插入数据时要注意，数字数据可以直接插入，但是字符数据和日期数据要使用单引号（必须是英文半角输入状态下的单引号）引起来。如果是 Unicode 数据，应该在字符数据的引号前使用 N 字符（N 一定要大写）。

1）插入一条记录

插入一条记录的语法格式如下：

```
INSERT
```

```
INTO [database_name. ] < table_name > [ ( < column_name > [ , … n ] ) ]
VALUES (< constant > [ , … n ])
```

各参数说明如下。

(1) database_name:指定向哪个数据库插入数据,如果省略,即指当前连接的数据库。

(2) table_name:指定向哪个表插入数据。

(3) column_name:表中的列名,当指定 VALUES 的全部数据时可省略;如果指定了 column_name,则没有出现在子句中的 column_name 将被取 NULL。

(4) constant:插入的数据值。

例如,向 student 数据表插入一条记录,INSERT 语句如下:

```
INSERT INTO student(sno, sname)
VALUES ('201125011001', N'许宏')
```

如果 INTO 子句中没有任何列名,则 VALUE 子句后的列值顺序必须与表结构的列顺序一致。

2) 插入子查询结果

插入子查询结果的语句可以将多条满足条件的记录添加到目的表中,即一次插入多条记录。插入子查询结果的语法格式如下:

```
INSERT
INTO [database_name. ] < table_name > [ (< column_name > [ , … n ] ) ]
SELECT < select_list >
```

各参数说明如下。

(1) database_name:指定向哪个数据库插入数据。

(2) table_name:指定向哪个表插入数据。

(3) column_name:表中的列名。

(4) select_list:SELECT 查询结果。

例如,假设已建立 sc_2011 表,其表结构与 sc 表结构一致。将学号为 201125011001 的学生的各科成绩添加到 sc_2008002 表中,插入语句如下:

```
INSERT INTO sc_2011
SELECT sno, cno, grade
FROM sc
WHERE sno = '201125011001'
```

需要注意的是,标识字段可以设定为自动加 1,此时,在插入记录时不能为标识字段赋值。还要注意数据表的结构定义,如果有些字段不能为空,则插入记录时必须指定取值。

9. 修改数据记录

可以使用 UPDATE 语句更新表中已经存在的数据。UPDATE 语句可以一次更新一行数据、多行数据,甚至可以一次更新表中的全部数据行。UPDATE 语句的语法格式如下:

```
UPDATE [ database_name. ] < table_name >
SET {< column_name > = < expression >}[ , … n ]
```

```
[WHERE <search_condition>]
```

各参数说明如下。

（1）database_name：指定修改的数据所属的数据库。

（2）table_name：指定修改的数据所属的表。

（3）column_name：表中的列名。

（4）expression：用于取代相应的属性列值的表达式值。

（5）search_condition：指定修改条件，即修改符合条件的列值。若省略 WHERE 子句，则将更新表中所有记录。

【例 6-17】 修改 student 表中的数据，将 201125011001 号学生的名字改为"许小雅"，性别改为"女"，成绩加 10 分。代码如下：

```
UPDATE student
SET sname = N'许小雅', sname = N'女',grade = grade + 10
WHERE sno = '201125011001'
```

10. 删除数据记录

使用 DELETE 语句可以从表中删除一行或多行记录。DELETE 命令的语法如下：

```
DELETE FROM [database_name.] <table_name>
[WHERE <search_condition>]
```

各参数说明如下。

（1）database_name：指定删除记录的表所属的数据库。

（2）table_name：指定删除的记录所属的表。

（3）search_condition：指定删除条件，即删除符合条件的记录。若省略 WHERE，则默认删除所有记录。

例如，要删除 student 数据表中的 2008 级所有学生的记录，语句为：

```
DELETE FROM student
WHERE sno LIKE '2008[0-9][0-9]%'
```

假设学号的编码规则是：年级（4 位）＋ 专业（2 位）＋序号（多位），条件中的序号位数不确定。

6.5.3 数据库操作

在 Web 应用开发中，通常需要对数据库进行操作，包括数据库的查询、修改、插入、删除等。对数据库的操作，可以直接在 JSP 页面中完成，也可以将数据库操作编写成一个 JavaBean，然后在 JSP 页面中调用。

1. 数据库操作的基本步骤

对数据库的操作，可以分成以下四个基本步骤。

(1) 加载数据库驱动程序:

```
String sDBDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
try {   // 加载数据库驱动程序(Microsoft SQL Server 2005)
        Class.forName(sDBDriver).newInstance();
}
```

(2) 连接数据库服务器:

```
String strConn = "jdbc:sqlserver://localhost:1433;DatabaseName = gsl_messageboard";
String strUser = "sa";
String strPassword = "sa";
Connection conn = null;
conn = DriverManager.getConnection(strConn, strUser, strPassword);
```

(3) 创建 Statement 对象,并设置数据集游标类型与操作权限。

在执行具体的数据库 SQL 命令以前,需要创建 Statement 对象,并设置数据集游标类型与操作权限。

```
private java.sql.Statement stmt = null;
Statement stmt = conn.createStatement("游标类型", "记录更新权限");
```

① 可设置的游标类型:

- ResultSet.TYPE_FORWORD_ONLY,只可以向前移动。
- ResultSet.TYPE_SCROLL_INSENSITIVE,可卷动,不受其他用户对数据库更改的影响。
- ResultSet.TYPE_SCROLL_SENSITIVE,可卷动,当其他用户更改数据库时这个记录也会改变。

② 记录更新权限:

- ResultSet.CONCUR_READ_ONLY,只读。
- ResultSet.CONCUR_UPDATABLE,可更新。

例如,要对数据表进行查询操作,语句为:

```
Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                      ResultSet.CONCUR_READ_ONLY);
```

(4) 数据的查询、插入、修改与删除。当 stmt 对象创建后,即刻调用 Statement 对象的 executeQuery()方法和 executeUpdate()方法,利用 SQL 命令,来实现数据库数据表的查询、插入、修改与删除操作。

2. 数据库查询操作

在数据库中查询数据,是通过 Statement 对象的 executeQuery()方法实现的,运行结果返回一个 ResultSet 数据集。一般形式如下:

```
ResultSet rs = stmt.executeQuery(strSQL);
```

其中,strSQL 为一个 SELECT 形式的 SQL 命令,具体命令根据程序的功能书写。命令执行结果为查询结果数据集。然后可以通过 rs.getString()等方法来取得各个列的数据。

1）在结果数据集中定位数据记录

如果 ResultSet 是可卷动的，可以使用下列 ResultSet 方法来定位数据记录。

（1）boolean rs. absolute(int row)：绝对位置，负数表示从后面数。

（2）void rs. first()：将指针移动到结果集对象的第一行。

（3）void rs. last()：将指针移动到结果集对象的最后一行。

（4）boolean rs. previoust()：将指针移动到当前行的前一行。

（5）boolean rs. next()：将指针移动到当前行的下一行。

（6）void rs. beforeFirst()：将指针移动到结果集对象的头部，即第一条之前。

（7）void rs. afterLast()：将指针移动到结果集对象的末尾，即最后一条记录之后。

（8）boolean rs. isFirst()、boolean rs. isLast()、boolean rs. isBeforeFirst()、boolean rs. isAfterLast：用于判断当前的位置。

需要特别说明的是，刚打开数据表时，处于第一条记录之前。

2）获得数据库数据

通过 ResultSet 中的 get 方法可以取得数据表中当前记录的相应列值。常用的方法见表 6-9。

表 6-9　从 ResultSet 中获取数据的方法

SQL 类型	说　明	JSP 类型	ResultSet 方法
nchar、nvarchar、ntext	Unicode 字符串数据类型	String	String getString(col)
char、varchar、text	非 Unicode 字符串数据类型	String	String getString(col)
binary、varbinary	二进制字符串数据类型、二进制字符串数据类型	byte[]	byte[] getBytes(col)
bit	整数数据类型，取值为 1、0 或 NULL	Boolean	boolean getBoolean(col)
tinyint	1B	Integer	byte getByte(col)
smallint	2B	Integer	short getShort(col)
integer	4B	Integer	int getInt(col)
bigint	8B	Long	long getLong(col)
decimal[(m[,d])]	定点小数数据类型，m 是十进制数字的总个数，d 是小数点后面的数字个数	String	String getString(col)
real(m,d)	单精度浮点型，8 位精度(4B)	Float	float getFloat(col)
float(m,d)	双精度浮点型，16 位精度(8B)	Double	double getDouble(col)
datetime	日期和时间数据类型	String java. util. Date Date	String getString(col) java. sql. Date getDate(col)
money、smallmoney	货币数据类型，默认为 2 位小数的 decimal 类型。money 占 8B，smallmoney 占 4B	String	String getString(col)

在上述关于数值的类型中，decimal(m,d) 为定点小数类型，定点类型在数据库中存放的是精确值。参数 m 是定点类型数字的最大个数(精度)，范围为 0～65，d 是小数点右侧数

字的个数,范围为 0～30,但不得超过 m。对定点数的计算能精确到 65 位数字。

浮点型小数分为单精度 float 和双精度 double 型,参数 m 只影响显示效果,不影响精度,d 却不同,会影响到精度。比如设一个字段定义为 float(5,3),如果插入一个数 123.45678,实际数据库中存的是 123.457,小数点后面的数被四舍五入截成 457 了,但总个数不受到限制(6 位,超过了定义的 5 位)。因此,在定义类似价格等精确数据时,使用浮点型小数时需要特别注意。

对于 SQL 中的 Datetime 类型列值,在 JSP 中可以通过 getString()取得对应的日期数据,并在日期和 String 之间自动转换。对于 decimal、money 和 smallmoney 等数值型数据字段,ResultSet 类中没有对应的 getXXX 方法读取数据表字段的值,可以使用 getString()返回一个 String 对象,然后在 String 和数值间转换。

对于数据库中 longvarchar 和 longvarbinary 类型的数据字段,应进行流操作,介绍略。

3. 更新数据库

对数据库的更新,可以采用下述方法。

(1) stmt.executeUpdate("strSql")方法:strSql 是一条 SQL 更新语句。执行 UPDATE、INSERT 和 DELETE 操作,返回影响到的条数。用户通过返回值,可以判断数据库操作是否成功。

例如:试图插入一条记录,而该记录的主关键字已经存在于数据表中,此时则返回 0,即插入不成功。灵活使用主关键字定义可以避免插入重复记录,而不必在插入前首先查询记录是否存在,节省操作时间。

(2) stmt.execute()方法:在不知道 SQL 语句是查询还是更新的时候使用该方法。如果产生一条以上的对象时,返回 true,此时可用 stmt.getResultSet()和 stmt.getUpdateCount()来获取 execute 结果,如果不返回 ResultSet 对象则返回 false。

除了 Statement 的 executeUpdate 之外还可以用 ResultSet 来更新数据库。例如:

```
rs.updateInt(1,10);
rs.updateString(2,"xxx");
rs.updateRow();
```

4. 使用预编译 PreparedStatement

PreparedStatement 对象和 Statement 对象类似,都可以用来执行 SQL 语句。但是,通过 PreparedStatement 对象执行 SQL 语句的速度更快。因为,数据库会对 PreparedStatement 的 SQL 语句进行预编译,而且仍旧能输入参数并重复执行编译好的查询速度比未编译的要快。

例如:

```
PreparedStatement stmt = con.preparedStatement("Insert Into users(userid, username)
values(?,?)");
stmt.clearParameters();
stmt.setInt(1,2);
stmt.setString(2,"xxx");
stmt.executeUpdate();
```

6.5.4　数据库编程举例

JSP 工作在三层结构的中间层，主要的功能是接收客户的请求，从数据库服务器获得信息，然后以 HTML 的形式返给客户浏览器。下面举例说明在 JSP 中访问 Microsoft SQL Server 2005 数据库的过程。

【例 6-18】　编写一个 JSP 页面，完成对网站留言板数据的操作，包括留言、查看留言、删除留言等。

分析：在 Web 系统中，通常设置留言板功能，方便用户对站点使用中遇到的问题或用户的意见和建议进行留言，从而利于管理员对站点进行维护和改进。

具体操作步骤如下。

(1) 创建留言板数据表 msgboard。设在 Web 应用系统中有一个留言板，供用户对站点进行留言，管理员可以查看用户留言并回复，用户可以查看留言及站点管理员对留言的回复，也可以删除自己的留言。留言板数据存储在站点系统数据库 gslpub 的 msgboard 数据表中。

留言板数据表 msgboard 的结构定义如图 6-8 所示。

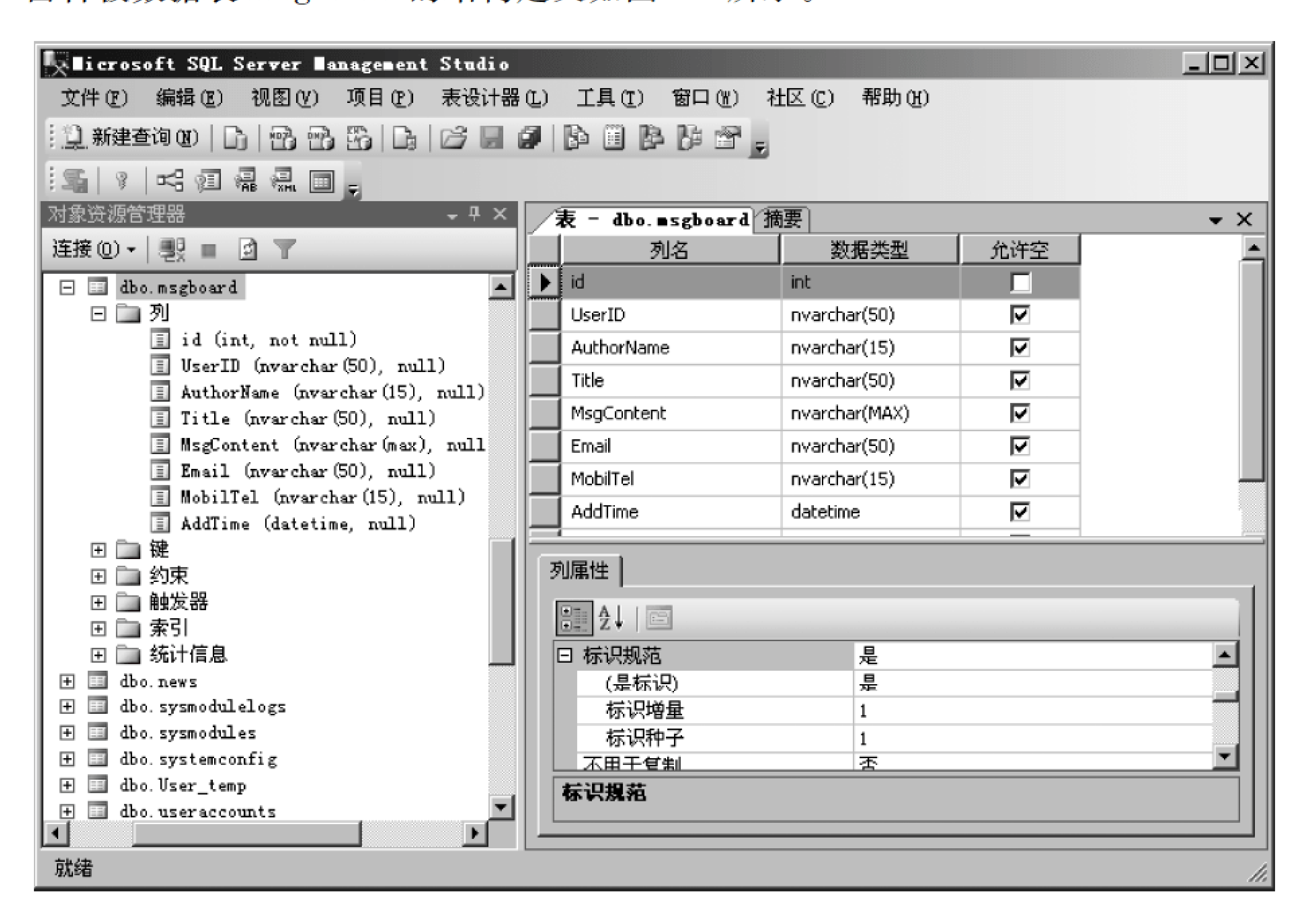


图 6-8　留言板数据表 msgboard 的结构定义

在 msgboard 数据表中，不设置主关键字，但定义了标识字段 id 为 int，并且设标识规范为“是”，标识增量为 1。在数据表中，标识字段的功能和主关键字类似，也可以唯一地定为一条记录，但标识字段的值不需要用户输入或导入，它的值自动生成。主关键字的取值必须由用户输入，不能为空，如果要导入数据表，也需要为每条记录给定主关键字的值。

(2) 为了便于代码的重用，我们定义一个数据库操作的 JavaBean。在 WEB-INF\

classes\pub 文件夹下，创建一个名为 db_gslpub.java 的文件，定义一个 JavaBean，封装有关数据库 gslpub 操作中的数据库连接、查询、数据更新以及断开数据库连接操作。

代码清单：

```
package pub;
import java.sql.*;
import java.text.*;
import java.io.*;
import java.util.*;
//////////////////////////////////////////////////////////////////////////////////////
// 连接 gslpub 数据库
public class db_gslpub
{
    String sDBDriver = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
    String sConnStr = "jdbc:sqlserver://localhost:1433;DatabaseName = gslpub";
    String strUser = "sa";
    String strPassword = "sa";
    Connection conn = null;
    private java.sql.Statement stmt = null;
    public ResultSet rs = null;
    // 默认构造函数,加载 JDBC 驱动程序
    public db_gslpub()
    {
      try {
          Class.forName(sDBDriver).newInstance();
      }
      catch(Exception e) {
          System.err.println("gslpub conn(): " + e.getMessage());
      }
    }
    // 定义数据库查询方法,返回查找的数据集
    public ResultSet executeQuery(String sql)
    {
      rs = null;
      try {
          conn = DriverManager.getConnection(sConnStr, strUser, strPassword);
          Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                                ResultSet.CONCUR_READ_ONLY);
          rs = stmt.executeQuery(sql);
      }
      catch(SQLException ex) {
            System.err.println("aq.executeQuery: " + ex.getMessage());
      }
      return rs;
    }
    // 定义修改数据库的方法: 插入、删除、修改操作
    public int executeUpdate(String sql)
    {
        int returnVal = -999;
        try {
```

```
            conn = DriverManager.getConnection(sConnStr, strUser, strPassword);
            Statement stmt = conn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                                  ResultSet.CONCUR_UPDATABLE);
            returnVal = stmt.executeUpdate(sql);
          }
        catch(SQLException ex) {
              System.err.println("aq.executeUpdate: " + ex.getMessage());
        }
        return returnVal;
    }
    // 断开数据库连接
    public void disconnectToDB() throws java.sql.SQLException
    {
        if (rs! = null) {
          rs.close();
          rs = null;
        }
        if (stmt! = null) {
          stmt.close();
          stmt = null;
        }
        if (conn! = null) {
          conn.close();
          conn = null;
        }
    }
}
```

在 Web 开发中，不建议在 JSP 中编写数据库的访问程序。对数据库的访问通常封装在一个 JavaBean 中来完成，这样可以更好地实现业务逻辑和 JSP 中 HTML 代码的分离，同时也便于系统的维护，例如数据库升级后，原先的数据库访问语句可能不兼容。

在这个 JavaBean 中，在构造函数中完成 JDBC 驱动的加载，另外定义了两个公有(public)方法来操作数据库，即进行数据记录的查询和更新。最后是一个关闭数据库连接的方法。

当上述文件编辑完成后，将 db_gslpub.java 文件保存，编译，则在 Web 应用根目录下的 WEB-INF\classes\pub 文件夹下生成 db_gslpub.class 文件，该 JavaBean 的.class 文件即可在 JSP 页面中被调用。

(3) 在 JSP 中使用 JavaBean。下面是一个留言板浏览页面，文件名为 msgboard-list.jsp，其功能是分页显示留言数据表中的留言记录。

代码清单：msgboard-list.jsp

```
<%@ page contentType="text/html;charset=gb2312" %>
<%@ page import="java.sql.*" %>
<%@ page import="java.*" %>
<jsp:useBean id="gslpub" class="pub.db_gslpub" scope="page"/>
<%!
int intpagesize=10,introwcount,intpagecount,intpage,i;
String strpage;
```

```
ResultSet rs = null;
%>
<%
strpage = request.getParameter("page");
if(strpage == null)
   intpage = 1;
else{
   intpage = java.lang.Integer.parseInt(strpage);
   if(intpage < 1)
       intpage = 1;
}
%>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312" />
<link rel = "stylesheet" type = "text/css" href = "pubcss/mycommon.css">
<link rel = "stylesheet" type = "text/css" href = "pubcss/mytable.css">
</head>
<body>
<table class = "table-hasframe" width = "750" cellpadding = "0" cellspacing = "0" align =
"center">
<tr height = "40">
<td class = "table-topic" width = "100%" colspan = "8">【留言板】用户留言列表</td>
<tr>
<tr height = "35">
    <td class = "cell-title" width = "5%">序号</td>
    <td class = "cell-title" width = "45%">标题</td>
    <td class = "cell-title" width = "20%">日期</td>
    <td class = "cell-title" width = "15%">留言人</td>
    <td class = "cell-title" colspan = "3">操作</td>
</tr>
<%
try
{
   String id,title,msgcontent,addtime,author;
   String href1,href2,href3;
   rs = gslpub.executeQuery("SELECT * FROM  msgboard ORDER BY AddTime DESC");
   if(!rs.next())
   {
%>
<tr>
    <td class = "cell-title" width = "100%" colspan = "8">留言板空</td>
</tr>
<%
   }
   else
   {
       rs.last();
       introwcount = rs.getRow();
       intpagecount = (introwcount + intpagesize - 1)/intpagesize;
       if(intpage > intpagecount) intpage = intpagecount;
```

```
        if(intpagecount > 0)
        {
           rs.absolute((intpage - 1) * intpagesize + 1);
           i = 0;
           while(i < intpagesize && !rs.isAfterLast())
           {
               id = rs.getString("id");
               title = rs.getString("title");
               addtime = rs.getString("addtime");
               author = rs.getString("authorname");
               href1 = "msgboard - detail.jsp?id = " + id;
               href2 = "msgboard - replay.jsp?id = " + id;
               href3 = "msgboard - delete.jsp?id = " + id;
%>
<tr height = "30">
     <td class = "cell - normal"><% = (intpage - 1) * intpagesize + 1 + i %></td>
     <td class = "cell - normal"><a href = "<% = href1 %>"><% = title %></a></td>
     <td class = "cell - normal"><% = addtime %></td>
     <td class = "cell - normal"><% = author %></td>
     <td class = "cell - normal"><a href = "<% = href1 %>">查看</a></td>
     <td class = "cell - normal"><a href = "<% = href2 %>">回复</a></td>
     <td class = "cell - normal"><a href = "<% = href3 %>" onclick = "{if(confirm('确定要删除
吗?')){return true;}return false;}" target = "_self">删除</a></td>
</tr>
<%
               rs.next();
               i++;
           }
        }
%>

<tr height = 30>
     <form name = "form1" method = "post" action = "msgboard - list.jsp">
     < td class = "cell - title" colspan = "8">第<% = intpage % > 页     共<% =
intpagecount %>页
<%
     if(intpage>1)
     {
%>
       <a href = "message_list.jsp?page = 1">第一页</a>   
       <a href = "message_list.jsp?page = <% = intpage - 1 %>">上一页</a>  
<%
     }
     if(intpage < intpagecount)
     {
%>
       <a href = "message_list.jsp?page = <% = intpage + 1 %>">下一页</a>  
       <a href = "message_list.jsp?page = <% = intpagecount %>">最后一页</a>
<%
      }
%>
```

```
        <input type="text" name="page" value="0" style="height:17px;width:20px;">
        <input type="submit" value="转到" style="height:18px;">
      </td>
      </form>
    </tr>
    <%
       }//else end
    }//try end
    catch(Exception ex){
        out.print(ex.getMessage());
    }
    finally{
        rs=null;
        gslpub.disconnectToDB();
    }
    %>
    </table>
    </body>
    </html>
```

上述代码演示了数据库在JSP中的简单分页浏览，页面中引用了用户自定义的样式表文件，定义了一组关于<table>和<td>标记的样式类。使用样式类来规范Web页面的显示是良好的习惯。

在浏览器中，运行msgboard-list.jsp页面，显示结果如图6-9所示。



图6-9 数据库浏览列表界面

上述界面是我们在许多的Web应用中见到的用户界面。在表格中的每一项的后面都有一个“删除”超链接，单击该超链接，可以删除当前记录。对应该超链接的代码是：

```
<a href = "msgboard - delete.jsp?id = <% = id %>"
onclick = "{if(confirm('确定要删除吗?')){return true;}return false;}"
target = "_self">删除</a>
```

针对上述代码，说明如下。

(1) 如果超链接中有 onclick 事件，则单击超链接时，首先执行 onclick 对应的函数，如果函数返回 true，则接下来转到 href 设置的页面，如果函数返回 false，则不转到 href 指定的页面。

(2) 每一个超链接，都会打开一个窗口，设定 target="_self"，可以保证打开的窗口覆盖当前窗口。这样在 msgboard-delete.jsp 中，删除记录后，再通过 response.sendRedirect("msgboard-list.jsp");来显示记录列表，产生页面刷新的效果，即可看到删除后的结果。

删除记录对应的 msgboard-delete.jsp 页面，代码清单如下：

```
<%@ page contentType = "text/html;charset = GBK" %>
<%@ page import = "java.sql.*" %>
<%@ page import = "java.text.*" %>
<jsp:useBean id = "gslpub" class = "pub.db_gslpub" scope = "page"/>
<%
 String id = request.getParameter("id");
 if(id == null)//无内容则设为空串
     id = "";
String sqlString = "DELETE FROM msgboard WHERE id = '" + id.trim() + "'";
try {
    gslpub.executeUpdate(sqlString);
}
catch(Exception ex) {
    out.print(ex.getMessage());
}
finally{
    gslpub.disconnectToDB();
    response.sendRedirect("msgboard - list.jsp");
}
%>
```

删除记录页面是一个纯粹的 JSP 程序，没有要显示的内容，删除成功后，转回 msgboard-list.jsp 页面，即可看到记录被删除的结果，产生页面刷新效果。在使用 response.sendRedirect(url)命令时，如果 URL 中可以包含参数，由于 HTTP 中文编码问题，中文参数将不能正确传送，可以通过 session 和 application 来传递。

在书写 SQL 命令串时，特别是 SQL 中关键字之间的空格容易忽视，书写格式一定要清晰，以便于阅读和修改。例如，修改记录的 SQL 命令可以书写为：

```
String strSQL = "UPDATE msgboard SET "
            + "AuthorName = N'" + 珍妮 + "',"
            + "Email = '" + jane@sdu.edu.cn + "',"
            + "MobilTel = '" + 135xxxx5123 + "'"
            + "WHERE UserID = '" + jane + "'";
```

在执行修改命令以前，可以用下面的语句来显示上述的 SQL 命令串是否正确：

```
out.println(strSQL + "<br>");
```

6.6 综合举例

在 Web 系统的开发过程中,虽然每个系统的业务领域可能不同,但许多业务模块的程序实现是相似的。本节选取了笔者在项目研发中遇到的一些公共功能模块作为例子讲解,一则是为了练习已学的知识,其次则是为了避免为了举例而举例,举例内容缺少实用性。通过来自项目研发中的例子,不仅能更好地讲解所学的书本知识,还可以将这些例子代码应用到实际项目的研发中,为以后的项目研发提供借鉴和帮助。

6.6.1 文件上传操作

在许多 Web 应用中,都会用到文件和文件夹操作,例如论文管理、作业提交等。在基于 Java 的技术中,有关文件和文件夹的操作被封装在 java.io 包中,下面举例说明 JSP 中文件(夹)的创建、删除以及文件的复制等操作的实现。

【例 6-19】 有一新闻公告发布页面,页面中允许用户上传附件文件,并规定附件文件的类型,上传的附件数量不限,在整个新闻公告确定发布前,可以删除已经上传的附件。编写相应的 JSP 代码,实现文件上传功能。

分析:由于可以上传多个文件,并且在新闻公告确认发布前,允许删除上传的附件,因此该页面还应该提供一个显示上传附件的目标 iframe,作为 form 表单的 target 输出。界面设计如图 6-10 所示。

【添加新闻公告】
信息类型:新闻公告
标题:
详细内容
字体 大小 源代码
输入正文
上传文件: 附件标题: 附件类型: 插图 附件
文件名: 浏览... 上传
作者:郝兴伟 部门:中心办公室
提 交 取 消

图 6-10 新建新闻公告界面

在上述界面设计中，在“上传文件”区域，每次上传一个文件，需要输入附件标题；单击“浏览”按钮，选择上传的文件；然后单击“上传”按钮，则将文件上传到 Web 服务器，此时在下面的文本框中显示上传的文件列表。重复执行上述过程，可以上传多个附件。

在上传区域的文件列表中，每个上传的文件后面都显示一个“删除”超链接，单击该超链接，上传的文件将从服务器上删除。

实现上述功能，需要三个 JSP 文件，一个是 form 页面，一个是 Web 服务器端保存文件页面，还有一个页面负责删除上传的文件。

(1) 文件上传表单页面 news-add.jsp。从上述的页面设计可见，由于要实现多个附件的文件上传，因此文件上传是一个表单。但是，新建一个新闻公告，还包含许多其他信息，例如标题、正文等，这些信息需要通过另外的表单来提交。本处置介绍文件上传涉及的代码部分。

代码清单： news-add.jsp 文件上传涉及的界面核心代码

```
<form name="form2" method="POST" action="news-addfilesave.jsp?delfileflag=1"
        enctype="multipart/form-data" target="filelistbox">
  <input type="hidden" name="newscode" value="<%=newscode%>">
  <input type="hidden" name="addfilelist" value="">
  <input type="hidden" name="addfiletitle" value="">
  <tr height="26">
     <td class="table_cell" rowspan="3" align="right">上传文件:</td>
     <td class="table_cell" align="left">
         附件标题:<input type="text" name="showaddfiletitle" size="47">
         附件类型:<input type="radio" name="addfileflag" value="插图">插图
                  <input type="radio" name="addfileflag" value="附件" checked>附件
     </td>
  </tr>
  <tr height="26">
       <td class="table_cell">  文件名:<span style="width:340px;overflow:hidden">
        <input name="tempstr" value="" size=60></span>
        <span style="width:80px;overflow:hidden">
        <input type="file" style="width:0px;margin-left:-4px" name="showaddfile"
onpropertychange="showfile(this)"></span>
        <input type="button" value="上传" onclick="Form2DataValid()">
       </td>
  </tr>
  <tr>
       <td class="table_cell" valign=center align="left">
          <iframe name="filelistbox" width="98%" height="45" style="border:0px solid #
FFFFFF"></iframe>
          </td>
    </tr>
</form>
```

在上述代码中，涉及两个客户端函数，即当用户单击“浏览”按钮时，显示用户选择的文件时用到的 showfile()，另一个为单击“上传”按钮时，执行的数据有效性验证和表单提交函数 Form2DataValid()。

代码清单：

```
////////////////////////////////////////////////////////////////////////////////
// 在使用< input type = 'file' />控件的过程中,不显示文件绝对路径,被 C:\fakepath\代替,避免
// 服务器获取用户端的真实路径,以增强安全性.用户可在浏览器中设置,显示真实路径
// 参数 obj 为 input file 对象
function getPath(obj)
{
    if(obj)
    {
        if (window.navigator.userAgent.indexOf("MSIE")>= 1)
        {
            obj.select();
            return document.selection.createRange().text;
        }
        if(window.navigator.userAgent.indexOf("Firefox")>= 1)
        {
            if(obj.files)
            {
                return obj.files.item(0).getAsDataURL();
            }
            return obj.value;
        }
        return obj.value;
    }
}
////////////////////////////////////////////////////////////////////////////////
// 显示文件真实路径和文件名
function showfile(obj)
{
    document.form2.tempstr.value = getPath(obj);
}
////////////////////////////////////////////////////////////////////////////////
// 表单 Form2 数据有效性验证、提交
function Form2DataValid(myform)
{
    // 附件标题验证
    var tmpstr = document.form2.showaddfiletitle.value;
    var tmpflag = false;
    if (tmpstr == "")
    {
        alert("附件标题不能为空!");
        document.form2.showaddfiletitle.focus();
        return false;
    }
    if (tmpstr.indexOf("&")>= 0 || (tmpstr.indexOf("#")>= 0)
    {
        alert("附件标题不能包含?、&、$ 、!、; 、# 等字符");
        document.form2.showaddfiletitle.focus();
        return false;
    }
```

```
    if (document.form2.showaddfile.value == "")
    {
        alert("请先选择文件");
        document.form2.showaddfile.focus();
        return false;
    }
    // 将附件文件列表存入 Form2 隐藏控件,从而上传到服务器端
    var iframestr = document.filelistbox.document.body.innerText;
    if (iframestr.length>0)
        document.form2.addfilelist.value = iframestr.substring(0,iframestr.indexOf("#"));
    else
        document.form2.addfilelist.value = "";
    document.form2.addfiletitle.value = document.form2.showaddfiletitle.value;
    // 清空文本框,以便填加下一个附件
    document.form2.showaddfiletitle.value = "";
    //file 的 value 是只读的,file 的 value 只能由用户输入和选择,不能解决清空本次输入的显示
效果
    document.form2.tempstr.value = "";

    document.form2.submit();
    return true;
}
```

(2) 表单处理程序页面 news-addfilesave.jsp。当单击“上传”按钮后,进行有效性验证,最后执行 form2.submit()操作,即提交表单 form2,此时 form2 的 action 参数指定的服务器处理程序 news-addfilesave.jsp 被执行,进行文件上传处理。

代码清单:

```
<%@ page contentType = "text/html;charset = GBK" %>
<%@ page import = "java.*" %>
<%@ page import = "java.util.*" %>
<%@ page import = "com.jspsmart.upload.*" %>
<jsp:useBean id = "mySmartUpload" scope = "page" class = "com.jspsmart.upload.SmartUpload" />
<jsp:useBean id = "mytime" scope = "page" class = "pub.mytime"/>
<jsp:useBean id = "gb2312" scope = "page" class = "pub.ISOtoGb2312" />
<jsp:useBean id = "workF" scope = "page" class = "pub.fliter" />
<%
String delfileflag = request.getParameter("delfileflag");
String newscode = "";
String addfileflag = "",addfiletitle = "",addfilelist = "";
String FileName = "",extname = "";
String iframestr = "";
String[] s1,s2;
// Initialization
mySmartUpload.initialize(pageContext);
// Only allow txt or htm files
mySmartUpload.setAllowedFilesList("htm,HTM,,doc,DOC,ppt,PPT,jpg,JPG,gif,GIF,rar,RAR");
// DeniedFilesList can also be used :
// mySmartUpload.setDeniedFilesList("exe,bat,jsp");
// Only allow files smaller than 50000 bytes
```

```
// mySmartUpload.setMaxFileSize(50000);
// Upload
mySmartUpload.upload();
// Save the files with their original names in a virtual path of the web server
try {
    //获取表单数据,与不含上传文件控件的 Form 获取方法 request.getParameter 方法不同
    Enumeration  enumer = mySmartUpload.getRequest().getParameterNames();
    while(enumer.hasMoreElements())
    {
        String  key = (String)enumer.nextElement();
        String[] values = mySmartUpload.getRequest().getParameterValues(key);
        if(key.equals( "addfileflag"))
        {
            addfileflag = values[0];
        }
        if(key.equals("addfiletitle"))
        {
            addfiletitle = values[0];
        }
        if(key.equals("newscode"))
        {
            newscode = values[0];
        }
        if(key.equals("addfilelist"))
        {   // 隐藏控件,存储已有的上传文件列表
            addfilelist = values[0];
        }
    }
  ///////////////////////////////////////////////////////////////////////////////
  //随机生成文件名
  String chose = "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ";
  char display[] = {'0','0','0','0','0','0'},ran[] = {'0','0','0','0','0','0'},temp;
  Random rand = new Random();
  for(int i = 0;i<6;i++)
  {
      temp = chose.charAt(rand.nextInt(chose.length()));
      display[i] = temp;
      ran[i] = temp;
   }
   String random_filename = String.valueOf(display);
   random_filename = workF.toHtml(random_filename);
   String save_filename = random_filename;
   ////////////////////////////////////////////////////////////////////////////////
   //保存上传文件
   com.jspsmart.upload.File myFile  =  mySmartUpload.getFiles().getFile(0);
   FileName  =  myFile.getFileName();
   extname  =  myFile.getFileExt();
   save_filename  +=  "."  +  extname;
   myFile.saveAs("/mynews/upload/"  +  newscode  +  "-" +  save_filename);
   ////////////////////////////////////////////////////////////////////////////////
   //输出附件列表,即一个 html 页面,其中页面中"#"以前的部分是页面间传递的数据
```

```
//设定的用于 split 的分割符
String f1 = "\10";
String f2 = "\20";
if ("".equals(addfilelist))
{
   addfilelist = addfileflag + f2 + addfiletitle + f2 + save_filename + f1;
 }
 else
 {
   addfilelist += addfileflag + f2 + addfiletitle + f2 + save_filename + f1;
 }
 iframestr = addfilelist + "#";
 // 输出 newscode + addfilelist,以"#"结尾的串,该输出在 form2 的 target 参数中输出
 out.print("<body style = 'margin - top:0px;margin - left:10px;font - size:12px;line - height:150 % '>");
 out.print("<span style = 'display:none;'>" + iframestr + "</span>");
 // 在 iframe 中输出附加文件列表,用户可以及时看到新加的附件列表
 s1 = addfilelist.split(f1);
 for (int i = 0;i < s1.length;i++)
 {
     s2 = s1[i].split(f2);
     out.print("[" + (i + 1) + "] " + s2[0] + ":" + s2[1] + "," + s2[2]);
     //addfilelist 参数不能在最后,否则其最后一个文件的分隔符"\10"将不能保存
     out.print("<a style = 'color:#FF0000;text - decoration:none' title = '删除上传的附件' href = 'news - addfiledelete.jsp?delfilestr = " + s1[i] + "&delfileflag = " + delfileflag + "&addfilelist = " + addfilelist + "&newscode = " + newscode + "'>");
     out.print("  &times;");
     out.print("</a>" + "<br>");
 }
 out.print("</body>");
}
catch (Exception e) {
    out.println("<b> Wrong selection : </b>" + e.toString());
}
%>
```

(3) 删除上传文件页面 news-addfiledelete.jsp。在上述页面中,对于已经上传的附件,列表中包含一个“删除”超链接,对应删除附件程序 news-addfiledelete.jsp。

代码清单:

```
<%@ page contentType = "text/html;charset = gb2312" %>
<%@ page language = "java" import = "java.util.*" pageEncoding = "gb2312" %>
<%@ page import = "java.io.*" %>
<jsp:useBean id = "gb2312" scope = "page" class = "pub.ISOtoGb2312" />
<%
String newscode = gb2312.convert(request.getParameter("newscode"));
String addfilelist = gb2312.convert(request.getParameter("addfilelist"));
String delfilestr = gb2312.convert(request.getParameter("delfilestr"));
String delfileflag = request.getParameter("delfileflag");
String[] s1,s2;
//要删除的文件信息
```

```
String f1 = "\10";
String f2 = "\20";
s1 = delfilestr.split(f2);
String delfilename = newscode + " - " + s1[2];
//求要删除的文件对应的串在总的串中的位置
delfilestr = delfilestr + f1;
int pos = addfilelist.indexOf(delfilestr);
if (pos > - 1)
{
   addfilelist = addfilelist.substring(0,pos) + addfilelist.substring(pos + delfilestr.length());
}
//如果所有文件被删空
String iframestr;
if (addfilelist.length()> 0)
   iframestr = addfilelist + "#";
else
   iframestr = "";
//输出新的 addfilelist,没有附件时只有 newscode,以便添加新的附件
out.print("<body style = 'margin - top:0px;margin - left:10px;font - size:12px;line - height:
150 % '>");
out.print("<span style = 'display:none;'>" + iframestr + "</span>");
//显示新的文件列表,如果没有附件,则只有 newscode,以便添加新的附件
if (addfilelist.length()> 0)
{
   s1 = addfilelist.split(f1);
   for (int i = 0;i < s1.length;i++)
   {
      s2 = s1[i].split(f2);
      out.print("[" + (i + 1) + "] " + s2[0] + ":" + s2[1] + "," + s2[2]);
      out.print("<a style = 'color: # FF0000;text - decoration:none' title = '删除上传的附件'
href = 'news - addfiledelete.jsp?delfilestr = " + s1[i] + "&delfileflag = " + delfileflag +
"&addfilelist = " + addfilelist + "&newscode = " + newscode + "'>");
      out.print("  &times;");
      out.print("</a>" + "<br>");
   }
}
out.print("</body>");
//在修改操作时,不真正的删除文件,只有"确认"再删除,此时从物理上删除文件
if ("1".equals(delfileflag))
{
   String mypath = request.getRealPath("");
   mypath += "\\mynews\\upload\\";
   File f = new File(mypath,delfilename);
   if(f.exists())
   {
      f.delete();//删除文件
   }
}
%>
```

关于新闻公告的其他信息处理,包括页面内容编辑、表单数据提交、数据库操作等将在

6.6.2 节介绍。

6.6.2 多表单数据处理

下面介绍内容输入表单 form1,和页面底部的"确定"、"取消"按钮。用于上传全部数据的表单 form3,还有一个单独用于"取消"按钮的表单,其目的是删除已经上载的附件。

1. 内容编辑用表单 form1 定义

在页面的顶部,是新闻公告内容编辑区域,使用了一个 FCKeditor(在线编辑)控件,和标准的 textarea(多行文本框)控件相比,它支持 html 格式,以及在内容中输入插图和图表,更加方便和强大。要引用 FCKeditor 控件,首先从网上下载该控件,复制到用户 Web 应用的根目录下,并进行相应的配置,然后在 news-add.jsp 中导入。代码如下:

```
<%@ page import = "net.fckeditor. * " %>
<%@ taglib uri = "http://java.fckeditor.net" prefix = "FCK"  %>
```

下面是 news-add.jsp 中 form1 表单涉及的相关代码。

代码清单:news-add.jsp 中 form1 表单定义界面核心代码

```
<form name = "form1" method = "post" action = "news - addsave.jsp">
<input type = "hidden" name = "newscode" value = "<% = newscode %>">
<input type = "hidden" name = "addfilelist" value = "">
<input type = "hidden" name = "author" value = "">
<input type = "hidden" name = "workunit" value = "">
<input type = "hidden" name = "myauthor" value = "">

<tr height = "40">
  <td class = "table_topic" colspan = "2" bgcolor = "#d6dff7">【添加新闻公告】</td>
</tr>
<tr height = "26">
   <td class = "table_cell" width = "10 % " align = "right">信息类型: </td>
   <td class = "table_cell" align = "left">
       <select name = "newstype" disabled>
<%
for (int i = 0;i < newstypeopt.length;i++)
{
 %>
     <option value = "<% = newstypeopt[i] %>" <% = selectedstr[i] %>><% = newstypeopt[i] %>
</option>
<%
}
 %>
     </select>
     </td>
</tr>
<tr height = "26">
     <td class = "table_cell" align = "right">标题: </td>
     <td class = "table_cell" align = "left">
```

```
            <input name="newstitle" type="text" size="90" maxlength="100">
        </td>
    </tr>
    <tr>
        <td class="table_cell" align="center">详<br>细<br>内<br>容</td>
        <td class="table_cell" align="left">
    <%
    net.fckeditor.handlers.PropertiesLoader.setProperty("connector.userFilesPath","/mynews/
    upload");
    net.fckeditor.handlers.PropertiesLoader.setProperty("connector.userFilesAbsolutePath","/
    mynews/upload");
     %>
        <FCK:editor instanceName="newscontent" width="100%" height="300">
      <jsp:attribute name="value">输入正文</jsp:attribute>
          <jsp:body>
              <FCK:config CustomConfigurationsPath="d:/GSL5.0/teacher/teacher.config.js" />
          </jsp:body>
        </FCK:editor>
        </td>
    </tr>
    </form>
```

在 form1 中,没有表单提交按钮,form1 的 action 属性设置了处理程序,但不会被执行。它是通过后面的 form3 中的“确定”按钮提交的。其输入数据将在页面底部的 form3 中通过“提交”按钮统一提交,如果单独提交,将出现程序运行不稳定的错误,也就是说,有时候数据提交成功,有时候会失败。

2. 表单 form3 定义及多表单数据的提交

表单 form3 包含“确定”和“取消”按钮,以实现整个页面数据的提交或放弃。form3 定义及核心代码如下。

代码清单: news-add.jsp 中 form3 表单定义界面核心代码

```
<form name="form3" method="post">
<tr height="26">
    <td class="table_cell" align="right">作者:</td>
    <td class="table_cell" align="left">
         <input name="author" type="text" value="<% =truename%>" size="20">
           部门:<input name="workunit" type="text" value="中心办公室" size="20">
    </td>
</tr>
<tr height="30">
   <td class="table_cell" colspan="2" align="center">
   <input type="button" value="提交"  class="mybutton" onclick="form1Submit()"> 
   <input type="button" value="取消"  class="mybutton" onclick="myFormCancel()">
   </td>
</tr>
<input type="hidden" name="myauthor" value="<% =truename%>">
</form>
<form name="formcancel" action="news-addcancel.jsp?" method="post">
```

```
    <input type="hidden" name="newscode" value="<% =newscode %>">
    <input type="hidden" name="addfilelist" value="">
</form>
```

在 form3 中定义了多个 hidden 元素，目的是将 form1 和 form2 表单数据复制到 form3 中，通过 form3 一并提交。此外，我们还定义了一个 formcancel 表达，对应“取消”按钮。单击“提交”按钮，执行 form123Submit()函数，对应的客户端脚本程序如下。

代码清单：news-add.jsp 中 form3 表单提交核心代码

```
////////////////////////////////////////////////////////////////////////////////
// form1 表单数据有效性验证
function form1DataValid()
{
   if (document.form1.newstitle.value == "")
   {
      alert("信息标题不能为空!");
      document.form1.newstitle.focus();
      return false;
   }
   if (document.form1.newscontent.value == "")
   {
      alert("信息内容不能为空!");
      return false;
   }
   return true;
}
////////////////////////////////////////////////////////////////////////////////
// 将 form2、form3 的数据复制到 form1 的 hidden 元素,通过 form1 的 action 属性提交到服务器
// 之所以使用 form1 提交而不用 form3 提交,是因为 form1 中包含 FCKeditor 元素,不好复制
// 到 form3
function form1Submit()
{
   if (!form1DataValid())
      return false;
   //  (1) 取 Form2 表单数据,即 iframe 中的文件列表,存储已有的附件信息
   var iframestr = document.filelistbox.document.body.innerText;
   if (iframestr.length>0)
   {
      document.form1.addfilelist.value = iframestr.substring(0,iframestr.indexOf("#"));
   }
   // (2)将 form3 数据复制到 form1,包括隐藏(hidden)控件数据
   document.form1.author.value =  document.form3.author.value;
   document.form1.workunit.value =  document.form3.workunit.value;
   document.form1.myauthor.value =  document.form3.myauthor.value;
   // (3)将 form1 中的所有 disable 输入元素取消 disable,以便于数据提交
   var len = document.form1.elements.length;
   var i;
   for (i=0;i<len;i++)
   {
      document.form1.elements[i].disabled = false;
```

```
    }
  document.form1.submit();
}
////////////////////////////////////////////////////////////////////////////////
// form3“取消”按钮处理函数,调用 formcancel.submit(),进而调用 formcancel 的 action
function myFormCancel()
{
  var iframestr = document.filelistbox.document.body.innerText;
  document.formcancel.addfilelist.value = iframestr.substring(0,iframestr.indexOf("#"));
  document.formcancel.submit();
}
```

在 form3 中定义了多个 hidden 元素,目的是将 form1 和 form2 表单数据复制到 form3 中,通过 form3 一并提交。此外,我们还定义了一个 formcancel 表单,对应“取消”按钮。表单“提交”对应的客户端脚本程序如下。

3. 表单数据的保存

当表单提交后,在服务器端执行 news-addsave.jsp 文件,读取用户的表单输入数据,并存储到相应的数据表中。部分核心代码清单如下。

代码清单: news-addsave.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ include file="../session-confirm.jsp" %>
<%@ page import="java.sql.*" %>

<jsp:useBean id="gslpub" scope="page" class="pub.db_gslpub" />
<jsp:useBean id="mytime" scope="page" class="pub.mytime"/>
<jsp:useBean id="gb2312" scope="page" class="pub.ISOtoGb2312" />
<jsp:useBean id="workH" scope="page" class="pub.CodeFilter" />
<%!
int retval;
%>
<%
String newsdate = mytime.getTime();
String ss = mytime.getcurrenttime();

String newscode = request.getParameter("newscode");
String newstitle = gb2312.convert(request.getParameter("newstitle"));
String newscontent = gb2312.convert(request.getParameter("newscontent"));
String addfilelist = gb2312.convert(request.getParameter("addfilelist"));

// 将 text 文本转换成 HTML 格式存储到数据库中
newscontent = newscontent.trim();
//newscontent = workH.toHtml(newscontent);

String strSQL;
strSQL = "INSERT INTO news(NewsCode,NewsTitle,NewsContent,AddfileList,NewsDate) VALUES
        ("+ "'" + newscode
        + "',N'" + newstitle
```

```
            + "',N'" + newscontent
            + "',N'" + addfilelist
            + "','" + newsdate
            + "')";

    try{
            retval = gslpub.executeUpdate(strSQL);
    }
    catch (Exception ex){
          out.print(ex.getMessage());
    }
    finally {
          gslpub.disconnectToDB();
    }
    %>
    <%
    if (retval == 1)
    {
    %>
    <html>
    <head>
    <meta HTTP-EQUIV="Content-Type" content="text/html; charset=gb2312">
    </head>
    <body bgcolor="#799ae1" leftmargin="0" topmargin="0">
    <table width="60%" border="1" cellpadding="0" cellspacing="0" bordercolor="#FFFFFF">
    <tr height="40">
      <td class="table_topic" align="center">信息添加成功</td>
    </tr>
    <tr height="77">
      <td class="table_cell">您已经成功添加了信息:<% =newstitle%></td>
    </tr>
    <tr height="30">
      <td class="table_cell">[<a href="news-add.jsp">继续添加</a>]</td>
    </tr>
    </table>
    </body>
    </html>
    <%
    }
    %>
```

在上述代码中,用到了几个 JavaBean,其中 ISOtoGB2312 负责汉字编码的转换,CodeFilter 用于将 HTML 中保留的一些符号(例如>、<等)进行转换,避免在页面输出这些符号时与 HTML 标记出现解析错误。这些代码的介绍略。

6.7 Web 系统的设计与开发

在本书的最后,我们将通过在线聊天程序的开发来对所学的知识进行综合的应用。在线聊天是许多网站都提供的一种功能,它包含服务端开发、客户端开发、AJAX 技术、JavaBean 以及数据库的应用,涉及的内容比较全面,相对容易理解。

6.7.1 系统分析

在线聊天程序系统中,用户可分为管理员用户和普通用户,普通用户可以选择聊天对象,发送聊天信息,管理员可以对在线用户发出警告,甚至将用户踢出聊天室。

聊天程序设计包括两个方面,一方面是客户端的用户界面,另一方面是服务端的数据库操作,用于记录用户的聊天记录。此外,为避免用户聊天信息更新带来的页面闪烁,我们采用 AJAX 技术,来进行客户端和服务端的异步数据传输。

6.7.2 系统设计

在系统分析完成后,接下来就是系统设计。系统设计可分为功能设计和数据库设计两个方面。功能设计就是根据需求分析,设计系统的整体框架、系统的各项功能,建立系统功能树。由于本系统功能很简单,详细的系统功能设计略。

聊天数据库记录聊天社区、聊天主题和聊天记录,包括三个数据表:社区数据表、主题数据表(房间)和聊天记录数据表。结构定义如下。

(1) 社区数据表 topic:社区数据表存储要进入的一级主题。结构定义如图 6-11 所示。

(2) 主题数据表(房间)second_topic:存储聊天主题,即一般的房间。结构定义如图 6-12 所示。

表 - dbo.topic 摘要

列名	数据类型	允许空
id	int	☐
topic_name	nvarchar(255)	☑
topic_note	nvarchar(255)	☑
key_word	nvarchar(255)	☑
		☐

图 6-11 社区数据表 topic 结构定义

表 - dbo.second_topic 摘要

列名	数据类型	允许空
id	int	☐
topic_id_1	nvarchar(255)	☑
topic_name	nvarchar(255)	☑
topic_note	ntext	☑
		☐

图 6-12 主题数据表(房间)second_topic 结构定义

(3) 聊天记录数据表 chat:记录聊天信息及系统公告。结构定义如图 6-13 所示。

表 - dbo.chat 摘要

列名	数据类型	允许空
id	int	☐
topic_id_1	char(10)	☑
topic_id_2	char(10)	☑
username	nvarchar(50)	☑
userid	nvarchar(50)	☑
userclass	nvarchar(50)	☑
to_user	nvarchar(50)	☑
color	nvarchar(50)	☑
face	char(10)	☑
note	nvarchar(MAX)	☑
addtime	nvarchar(50)	☑
addtime1	nvarchar(50)	☑
mode	char(10)	☑
		☐

图 6-13 聊天记录数据表 chat 结构定义

6.7.3　客户端页面设计

聊天程序主要涉及一个用户界面，这就是客户端用户界面。客户端用户界面通常分成几个区域：用户列表、聊天信息输入区域、聊天记录显示区域。上述用户界面可以通过 HTML 的表格和图层来布局。常见的用户界面如图 6-14 所示。



图 6-14　聊天系统客户端用户界面

聊天系统客户端界面分成三个区域，左侧为用户信息显示区域，显示用户个人信息和在线用户列表；右侧上部为聊天记录的显示区域，下部为用户输入区。下面介绍各个部分的实现代码，包括聊天主界面代码清单。

1. 客户端用户界面 chat-frame.jsp

聊天程序的主界面页面为 chat_frame.jsp，采用在 Table 中插入 iframe 方式，实现不同区域的功能。

代码清单： chat_frame.jsp

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*" %>
<%@ page import="java.sql.*" %>
<jsp:useBean id="gslpub" scope="page" class="pub.db_gslpub" />
<jsp:useBean id="mytime" scope="page" class="pub.mytime"/>
<%!
```

```
String useraccount = "",truename,nickname,userrole;
String topic_id_1,topic_id_2,note = "",dnow = "";
String sqlString = "";
%>
<%
topic_id_1 = request.getParameter("topic_id_1");
topic_id_2 = request.getParameter("topic_id_2");
session.setAttribute("topic_id_1",topic_id_1);
session.setAttribute("topic_id_2",topic_id_2);
useraccount = (String)session.getAttribute("useraccount");
truename = (String)session.getAttribute("truename");
nickname = (String)session.getAttribute("nickname");
if (nickname == null || "".equals(nickname))
    nickname = truename;
userrole = (String)session.getAttribute("userrole");
dnow = mytime.getTime();
try
{
    //(1)从数据库中清除 1 天以前的聊天记录
    String sqlString = "DELETE FROM chat WHERE addtime1! = '" + dnow.substring(0,10) + "'";
    gslpub.executeUpdate(sqlString);
    //(2)将系统通知插入到数据库中
    note = "『系统通知』: " + nickname + "加入了我们的讨论";
    sqlString = "INSERT INTO chatting(topic_id_1,topic_id_2,username,userid,userclass,
                              to_user,color,face,note,addtime,addtime1,mode) values( "
        + "'" + topic_id_1
        + "','" + topic_id_2
        + "',N'" + nickname
        + "','" + useraccount
        + "',N'" + userrole
        + "',N'" + "大家"
        + "','" + "#ff0000"
        + "',N'" + "无"
        + "',N'" + note
        + "','" + dnow
        + "','" + dnow.substring(0,10)
        + "','1')";
    gslpub.executeUpdate(sqlString);
}
catch(Exception ex){
   out.print(ex.getMessage());
}
finally{
   gslpub.disconnectToDB();
}
%>

<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312" />
</head>
```

```
<body>
<table id="main_table" width="100%" height="97%" cellpadding="0" cellspacing="0">
<tr>
   <td rowspan="2" width="250px">
       <iframe name=left frameBorder=0 scrolling=auto src="chat-left.jsp"></iframe>
   </td>
   <td id="right" height="70%">
  <iframe name=right frameBorder=0 scrolling=no src="chat-client.html"></iframe>
   </td>
</tr>
<tr>
   <td>
        <iframe name=bottom frameBorder=0 scrolling=no src="chat-input.jsp"></iframe>
   </td>
</tr>
</table>
<script>
this.moveTo(0,0)
this.resizeTo(screen.availWidth,screen.availHeight)
</script>
</body>
</html>
```

在上述代码中,为了保证界面的灵活性,引用了一个 css 文件 chat.css,代码从略。

2. 左侧信息列表 chat-left.jsp

左侧显示用户信息、在线用户列表等。

代码清单:

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*" %>
<%@ page import="java.sql.*" %>
<jsp:useBean id="gslpub" scope="page" class="pub.db_gslpub" />
<%!
String useraccount="",truename,nickname,userrole;
String topic_id_1,topic_id_2,topic_name;
String sqlString="";
ResultSet rs = null;
%>
<%
useraccount=(String)session.getAttribute("useraccount");
truename=(String)session.getAttribute("truename");
nickname=(String)session.getAttribute("nickname");
if (nickname==null || "".equals(nickname))
     nickname=truename;
userrole=(String)session.getAttribute("userrole");
topic_id_1=(String)session.getAttribute("topic_id_1");
topic_id_2=(String)session.getAttribute("topic_id_2").trim();

try {
```

```
        rs =  gslpub.executeQuery("SELECT * FROM chattopic2 WHERE id = '" + topic_id_2 + "'");
        rs.next();
        topic_name = rs.getString("topic_name");
    }
    catch(Exception ex) {
        out.print(ex.getMessage());
    }
    finally{
        rs = null;
        gslpub.disconnectToDB();
    }
    %>
<html xmlns = "http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv = "Content-Type" content = "text/html; charset = gb2312" />
<script language = "JavaScript">
////////////////////////////////////////////////////////////////////////////////
// 将在线用户列表中选择的聊天对象,显示在页面的 bottom 浮动帧中
function toperson(user)
{
   parent.bottom.form1.to_user.value = user;
   return false;
}
</script>
</head>

<body>

<table width = "93 % " cellpadding = "0" cellspacing = "0">
<tr>
    <td class = "row1"><b>在线信息</b></td>
</tr>
<tr>
    <td>
    <table class = "welcome" cellpadding = "0" cellspacing = "0" border = "0">
<%
sqlString = "SELECT username FROM chatting WHERE topic_id_1 = '" + topic_id_1 + "' AND topic_id_2
 = '" + topic_id_2 + "' GROUP BY username";
rs =  gslpub.executeQuery(sqlString);
rs.last();
int online_nums = rs.getRow();
 %>
<tr height = "20px">
    <td>当前在线 <font color = " #0033FF"><b><% = online_nums %></b></font> 人 </td>
</tr>
<tr height = "20px">
    <td>在线用户列表: </td>
</tr>
<%
rs.absolute(1);
String tousername;
```

```
do
{
   tousername = rs.getString("username");
 %>
<tr height = "20px">
   <td>
       <a href = "#" onClick = "toperson('<% = tousername %> ') " ><% = tousername %></a>
   </td>
</tr>
<%
} while(rs.next());
 %>
   </table>
   </td>
</tr>
</table>
<table class = "table" width = "93 % " cellpadding = "0" cellspacing = "0">
<tr>
     <td class = "row1"><b><a href = "chat - exit.jsp" target = '_parent>离开该话题</a></b>
</td>
</tr>
</table>
</body>
</html>
```

需要说明的是，当一个用户登录聊天界面时，系统发一个公告消息，同时将该用户存储在聊天数据表 chat 中，通过该数据表，可以显示在线用户列表。

在上述代码中，因为要显示用户列表，我们查询了整个数据表，如果仅仅是统计在线人数，可以使用下面的 SQL 指令：

```
sqlString  = "SELECT   COUNT(DISTINCT userid) AS nums "
        + "FROM     chatting "
        + "WHERE    (topic_id_1  =  '" + topic_id_1 +  "') AND (topic_id_2  =  "'+ topic_id_2 '"
rs = gslpub.executeQuery(sqlString);
if (rs.next())
  int online_nums = rs.getInt("nums");
```

上述 SQL 查询的结果集只包含一条记录，效率更高。因此，在数据库的操作中，要优化 SQL 指令，以提高服务器效率。

对于左侧最下面的"离开该话题"超链接，执行 chat-exit.jsp，和进入一个话题类似，在 chatting 数据表中插入一条广播消息记录，代码略。

此外，由于在线用户列表是动态变化的，不断地会有新的用户进入，也会有用户退出，因此用户列表需要实时更新以反映在线用户实际的情况。

3. 用户输入页面 chat_input.jsp

在聊天程序界面的右下部，是用户的输入区，输入信息后，同时将输入信息保存到聊天数据库中。在 chat-input.jsp 文件中，Form 表单的 action 属性设置为 chat-input.jsp 文件

本身,第一次连接到该文件时,即进入聊天室,只将公告存储到聊天数据表 chat 中。

输入页面 chat_input.jsp 关键代码清单如下:

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*" %>
<%@ page import="java.sql.*" %>
<jsp:useBean id="gslpub" scope="page" class="pub.db_gslpub" />
<jsp:useBean id="gb2312" scope="page" class="pub.ISOtoGb2312" />
<jsp:useBean id="mytime" scope="page" class="pub.mytime"/>

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<script language="javascript">
/////////////////////////////////////////////////////////////////////////////////
//验证数据的合法性
function form1submit()
{
   if (form1.content.value == "")
   {
      alert("请输入内容!");
      form1.content.focus();
      return false;
    }
}
</script>
</head>
<body>
<form name="form1" method="post" action="chat-input.jsp" onSubmit="return form1submit
()">
<table cellpadding="0" cellspacing="0" border="0">
<tr height="30px">
   <td style="text-align:left;margin-right:30px;">
     对象:<input class="pas" type="text" name="to_user" value="大家"/>  
    字体颜色:
      <select name='fontcolor' id='fontcolor' size="1" style="font-size:9pt">
      <option value='#3333ff' style='color:#3333ff'>默认</option>
      <option value='#000000' style='color:#000000'>黑色</option>
      <option value='#008800' style='color:#008800'>橄榄</option>
      </select>  
    动作与表情:<select name='mode' id='mode'>
    <option value='微笑着'>微笑</option>
    <option value='高兴地'>高兴</option>
    <option value='满意地'>满意</option>
    <option  value='痛苦地'>痛苦</option>
    </select>
    <font color="#0033FF">留言内容不要超过150字</font>
    </td>
</tr>
<tr height="60px">
```

```
    <td>内容:<textarea name="content" cols="95" rows="3" class="textarea"></textarea>
</td>
</tr>
<tr height="35px">
    <td align="center">
    <input type="submit" value="提交" class="menu">  
    <input type="reset" value="重置" class="menu">
    </td>
</tr>
</table>
</form>
</body>
</html>

<%!
String useraccount,truename,nickname="",userrole="";
String topic_id_1,topic_id_2,nowtime="",chat_record="";
String sqlString="";
%>

<%//接收客户端提交的数据
String to_user=(request.getParameter("to_user");
if(to_user==null) to_user="大家";
String fontcolor=request.getParameter("fontcolor");
if(fontcolor==null) fontcolor="";
String mode=request.getParameter("mode");
if(mode==null) mode="";
String content=codeToString(request.getParameter("content"));
if(content==null) content="";
useraccount=(String)session.getAttribute("useraccount");
truename=(String)session.getAttribute("truename");
nickname=(String)session.getAttribute("nickname");
if (nickname==null || "".equals(nickname)) nickname=truename;
userrole=(String)session.getAttribute("userrole");

if(!("".equals(content))) //如果输入内容不为空,则执行插入数据库操作
{
    chat_record=nickname+mode+"对"+to_user+"说: "+content;
    nowtime=mytime.getTime();
    sqlString=" insert into chatting(username,userid,to_user,color,face,note,addtime,
addtime1) values('"
            +"',N'" + nickname
            +"','" + useraccount
            +"',N'" + to_user
            +"','" + fontcolor
            +"','" + mode
            +"',N'" + chat_record
            +"','" + nowtime
            +"','" + nowtime.substring(0,13)
            +"')";
    gslpub.executeUpdate(sqlString);
}
%>
```

上述代码分为两个部分,第一部分是显示界面,第二部分的 JSP 程序用于存储聊天记录。若聊天内容输入为空,则不执行数据表插入记录操作。

4. 聊天信息显示 chat-client.htm

在聊天程序界面中,右上侧显示聊天信息,为避免刷新整个页面而引起的屏幕闪烁,采用 AJAX 技术,客户端代码为一个 html 页面 chat-client.htm。

代码清单:

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=GBK">
<script>
////////////////////////////////////////////////////////////////////////////////
// 滚动条置底
function scrollWindow()
{
   scroll(0, 100000);
   setTimeout('scrollWindow()', 200);
}
function convert(str)
{
   if (str! = "")
   {
    str = str.replace(/\r/g,"");
    }
    return str;
}
////////////////////////////////////////////////////////////////////////////////
// 创建 XMLHttpRequest 对象
var XMLHttpReq;
function createXMLHttpRequest()
{
   if(window.XMLHttpRequest)
   {  //Mozilla 浏览器
    XMLHttpReq = new XMLHttpRequest();
   }
   else if (window.ActiveXObject)
   {
      // IE 浏览器
      try      {
        XMLHttpReq = new ActiveXObject("Msxml2.XMLHTTP");
      }
      catch (e)  {
        try
        {
               XMLHttpReq = new ActiveXObject("Microsoft.XMLHTTP");
        }
    catch (e){}
      }
```

```
}
}
////////////////////////////////////////////////////////////////////////////////
//指定服务端程序
function sendRequest()
{
   createXMLHttpRequest();
   var url = "chat - server.jsp";
   XMLHttpReq.open("POST", url, true);
   XMLHttpReq.setRequestHeader("Content - Type","application/x - www - form - urlencoded");
   XMLHttpReq.onreadystatechange = processResponse;      //指定响应函数
   XMLHttpReq.send();                                    //发送请求

   scroll(0,100000);
   chat_content.scrollTop = chat_content.scrollHeight;
   setTimeout("sendRequest()",800);
}
////////////////////////////////////////////////////////////////////////////////
//处理返回信息函数,指定服务端输出显示的 id
function processResponse()
{
   if (XMLHttpReq.readyState == 4)
   {
     //判断对象状态
     if (XMLHttpReq.status == 200)
     {
        //信息已经成功返回,开始处理信息
        chat_content.innerHTML = convert(XMLHttpReq.responseText);
     }
     else
     {
        //页面不正常
         window.alert("您所请求的页面有异常.");
       }
     }
}
</script>
</head>
<body onload = "sendRequest()">
<div id = "chat_content" style = "width:750px; height:450px; margin - top:10px; background: #
FFFFFF; margin - left:10px;line - height:150 % ; overflow:scroll;">
</div>
</body>
</html>
```

现在,Web 开发人员更多地使用 div 和 css 来进行页面布局,div 可以起一个占位作用(定义图层的位置、高度和宽度),然后通过程序来动态地修改 div 中的内容,或者改变 div 的显示状态。在上述程序中,使用图层(id="chat_content")来定义聊天记录滚动显示区域。用户也可以通过 textarea 来定义显示区域,两者的不同是,在 div 定义中,可以格式化显示 html 文本,但 textarea 中,不能显示 html 格式化内容。

6.7.4 服务端程序设计

在服务端,主要完成聊天记录的数据存储和查找操作。主要涉及的程序就是用户输入中的服务端脚本部分,同时还涉及一个与客户端聊天信息显示进行异步传输的服务端程序,两者共同构成 AJAX 的异步传输,来避免用户界面的闪烁。

根据 AJAX 技术的工作机理,对应客户端的 chat-client.htm 文件,服务器端的处理程序为 chat-server.jsp,用于输出聊天记录。核心代码清单如下:

```
<%@ page contentType="text/html;charset=GBK" %>
<%@ page import="java.text.*" %>
<%@ page import="java.sql.*" %>
<jsp:useBean id="gslpub" scope="page" class="pub.db_gslpub" />
<%!
String topic_id_1="", topic_id_2="";
String color_text="",color_time="#999999",chat_content="",chat_time="";
%>
<%
topic_id_1=(String)session.getAttribute("topic_id_1");
topic_id_2=(String)session.getAttribute("topic_id_2");
ResultSet rs = null;
try
{
    //topic_id_1 为一级主题,topic_id_2 为二级主题,topic_id_1='0'且 topic_id_2='0'为系统公
告
    String strSQL="SELECT * FROM chatting "
                +"WHERE(topic_id_1='" + topic_id_1 + "'"
                +"AND topic_id_2='" + topic_id_2 + "')"
                +"OR(topic_id_1='0' and topic_id_2='0') "
                +"ORDER BY id";
    rs=gslpub.executeQuery(strSQL);
    while(rs.next())
    {
        color_text=rs.getString("color");
        chat_content=rs.getString("note");
        chat_time=rs.getString("addtime");
        out.print("<font color="+color_text+">"+chat_content+"</font> "+"<font
color="+color_time+">("+chat_time+")</font>");
        out.print("<br>");
    }
}//外层 try 结束
catch(Exception ex)
{
    out.print(ex.getMessage());
}
finally
{
    rs=null;
    gslpub.disconnectToDB();
}
%>
```

在 AJAX 技术中，服务端的 out 输出内容，被返回，然后显示在客户端页面的某个 div、span 或 td 元素内。

6.8　Java 开发工具简介

在计算机开发语言的历史中，从来没有哪种语言像 Java 那样受到如此众多厂商的支持，有如此多的开发工具。每一种 Java 开发工具各有所长，各有特点。下面对一些常见的 Java 开发工具进行简要介绍。

6.8.1　JDK

Java 开发工具包(JDK)是整个 Java 的核心，包括 Java 运行环境，Java 基础类库和一组建立、测试及建立文档的 Java 实用程序，这些实用程序包括开发用的 Java 编译器(javac)、Java 解释器(java)、打包工具(jar)、文档生成器(javadoc)、调试工具(jdb)等。不论什么 Java 应用服务器，实质都是内置了某个版本的 JDK，掌握 JDK 是学好 Java 的第一步。

最主流的 JDK 是 Sun 公司发布的 JDK，一般有三种版本，即 J2SE(标准版)、J2EE(企业版)和 J2ME，其中，J2SE 为一般用户常用的开发环境，J2EE 主要用于企业级 J2EE 应用程序，而 J2ME 则用于移动设备、嵌入式设备上的 Java 应用程序开发。不同的版本包含的包也不相同。除了 Sun 之外，还有很多公司和组织都开发了自己的 JDK，例如 IBM 公司开发的 JDK，BEA 公司开发的 Jrocket，还有 GNU 组织开发的 JDK，等等。其中 IBM 的 JDK 包含的 JVM 运行效率要比 Sun JDK 包含的 JVM 高出许多。而专门运行于 x86 平台的 Jrocket 在服务端运行效率也要比 Sun JDK 好很多。

JDK 简单易学，可以通过任何文本编辑器(如 Windows 记事本、UltrEdit、EditPlus、FrontPage 以及 Dreamweaver 等)编写 Java 源文件，然后在 DOS 状态下通过 javac 命令将 Java 源程序编译成字节码(.class 文件)，通过 Java 命令来执行编译后的 Java 文件，这能带给 DOS 时代程序员美好的回忆。

从初学者角度来看，采用 JDK 开发 Java 程序能够很快理解程序中各部分代码之间的关系，有利于理解 Java 面向对象的设计思想。但它的缺点也是非常明显的，就是从事大规模企业级 Java 应用开发非常困难，不能进行复杂的 Java 软件开发，也不利于团体协同开发。即便如此，JDK 仍然是许多 Java 专家最初使用的开发环境，尽管许多编程人员已经使用第三方的开发工具，但 JDK 仍被当做 Java 开发的重要工具。

6.8.2　Sun NetBeans 集成开发环境

NetBeans 是一个全功能的开放源码 Java IDE，可以帮助开发人员编写、编译、调试和部署 Java 应用，并将版本控制和 XML 编辑融入其众多功能之中。NetBeans 可支持 Java 2 平台标准版(J2SE)应用的创建、采用 JSP 和 Servlet 的 2 层 Web 应用的创建，以及用于 2 层 Web 应用的 API 及软件的核心组的创建。此外，NetBeans 还预装了一个 Web 服务器，即 Tomcat，从而免除了烦琐的配置和安装过程。所有这些都为 Java 开发人员创造了一个可扩展的开源多平台的 Java IDE，以支持他们在各自所选择的环境，如 Solaris、Linux、

Windows 或 Macintosh 中从事开发工作。

在 NetBeans 中,不仅可以开发 Java 程序,通过安装插件,也可以开发 C/C++ 程序。目前的版本是 NetBeans IDE 6.1,其中 NetBeans IDE 3.0 和 NetBeans IDE 5.0 都有中文版本,是目前进行大型 Web 应用开发的常用开发环境。用户可以从 Sun 的官方网站下载,网址是 http://www.sun.com/。

6.8.3 Eclipse 开发平台

2001 年 11 月,IBM 公司捐出价值 4000 万美元的源代码组建了 Eclipse 联盟,业界厂商合作创建了 Eclipse 平台。Eclipse 允许在同一 IDE 中集成来自不同供应商的工具,并实现了工具之间的互操作性。从本质上说,Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台,它只是一个框架和一组服务,用于通过插件组件构建开发环境,它附带了一个标准的插件集,包括 Java 开发工具(Java Development Tools,JDT)。

Eclipse 是一个 Java IDE,但 Eclipse 的目标不仅限于此。Eclipse 还包括插件开发环境(Plug-in Development Environment,PDE),这个组件主要针对希望扩展 Eclipse 的软件开发人员,因为它允许他们构建与 Eclipse 环境无缝集成的工具。Eclipse 是使用 Java 语言开发的,但它的用途并不限于 Java 语言。例如,支持诸如 C/C++、COBOL 和 Eiffel 等编程语言插件,Eclipse 框架还可用来作为与软件开发无关的其他应用程序类型的基础,比如内容管理系统。

基于 Eclipse 的应用程序的突出例子是 IBM 的 WebSphere Studio Workbench,它构成了 IBM Java 开发工具系列的基础。例如,WebSphere Studio Application Developer 添加了对 JSP、Servlet、EJB、XML、Web 服务和数据库访问的支持。Eclipse 是一款非常受欢迎的 Java 开发工具,用户越来越多。它的缺点是比较复杂,对初学者来说,理解起来比较困难。

本章小结

在 Web 开发中,服务端编程是 B/S 结构中最复杂的内容,它不仅要实现用户的业务逻辑,还包含大量的数据库操作,还必须保证数据库操作的效率和安全性。本章讲解了 Java 程序设计基础以及 JSP 技术两个方面,首先概要性地介绍了 Java 程序设计中涉及的概念,为 JSP 编程做好概念上的铺垫。在 JSP 技术中,以任务驱动的方式,讲解了服务端开发中遇到的共性问题及解决办法,包括 JSP 中的数据类型及其转换、数组、文件操作、JSP 内置对象、JSP 中的参数传递方法以及 JDBC 与数据库编程。这些内容是每一个 Web 应用中都可能遇到的,在讲解中给出了许多实用的代码和 JavaBean 类,这些代码都来源于我们的研发项目,读者可以直接应用在自己的项目开发中。

习题 6

一、简答题

1. 简述 Java 程序设计语言的特点。

2. 为什么说 Java 是一种完全面向对象的程序设计语言?

3. 在面向对象技术中，回答如下问题：

(1) 什么是类？什么是对象？简述它们之间的关系。

(2) 创建一个对象有哪些方法？

(3) 什么是类的构造函数？构造函数的一般功能是什么？

4. 列举主要的 Java 开发环境，它们的 JDK 相同吗？

5. 说明 Web 应用中的三层体系结构，并说明其优势。

6. 在 JSP 页面中，说明下列三条 page 指令的功能。

```
<%@ page import = "java.util.Date" %>
<%@ page errorPage = "errorPage.jsp" %>
<%@ page session = "true" %>
```

7. 在 JSP 中，有哪些常用的内置对象？简要说明它们的功能。

8. 简要说明 Web 应用的开发过程。

9. 在 JSP 中，在＜％！ … ％＞中声明变量和在＜％…％＞中声明变量有何不同？

10. 在字符串对象的操作中，使用 equals()方法判断两个字符串相等时，如何使用，为什么？使用 trim()方法来操作一个字符串变量时，如果发生异常，为什么？如何修改。

11. 在 JSP 中，有一个表单输入页面 myinput.htm，对应的表单处理页面为 myinputsave.jsp，如何在 myinputsave.jsp 中获取用户输入？

12. 在 Web 应用，页面之间的参数传递有哪些常用的方法？

二、编程题

1. 用户注册是大多数 Web 系统共有的功能。设计一个新用户注册功能，要求：

(1) 注册界面采用 htm，完成客户端的数据有效性验证。

(2) 在用户账户文本框后面，设计“检查用户名是否可用”超链接，采用 AJAX 技术编写相应的服务端数据库检测程序，检查完毕后，应显示账户是否可用信息。

2. 编写一个简单的作业提交系统，该系统具备以下简单功能：

(1) 用户注册功能，允许新用户注册。

(2) 作业提交功能，当用户首次登录后，在作业数据目录 homework 下建立一个以用户账户命名的子文件夹，存储该用户提交的作业。

(3) 作业管理功能，用户可以查看自己提交的作业，删除已经提交的作业，或者重新提交作业。

3. 写一个有关文件和文件夹操作的 JavaBean，封装常用的文件和文件夹(目录)操作，例如新建文件夹、新建文件、删除文件、删除文件夹中的所有文件、删除文件夹、复制单个文件、复制整个文件夹、移动文件到指定目录、移动文件夹到指定目录。

4. 在 Web 系统中，数据浏览、导入导出、打印等是许多系统的共有功能，编程实现下列功能：

(1) 任意设计一个数据表，分页显示数据库数据表中的数据记录。

(2) 编程实现将数据表导出为 Excel 表格。

(3) 编程实现页面的打印，以及生成 Word 文档。

5. 使用数据库技术和 JSP 开发一个简单的留言板系统。

参 考 文 献

1. 吴鹤龄，崔林. ACM 图灵奖——计算机发展史的缩影. 北京：高等教育出版社，2002.
2. 曾明. World Wide Web 开发使用指南. 北京：人民邮电出版社，1996.
3. Douglas E Comer. Internet 导引. 马志强，廖卫东译. 北京：清华大学出版社，1995.
4. 金勇华，曲俊生. Java 网络高级编程. 北京：人民邮电出版社，2002.
5. 孙卫琴. Java 面向对象编程. 北京：电子工业出版社，2006.
6. 夏先波. Java JDK 实例宝典. 北京：电子工业出版社，2007.
7. 林信良. Java JDK 6 学习笔记. 北京：清华大学出版社，2007.
8. 王国辉，王毅，尹相群. Java Web 开发技术方案宝典. 北京：人民邮电出版社，2008.
9. 袁建洲. JavaScript 编程宝典. 北京：电子工业出版社，2006.
10. 曹衍龙，叶达峰. AJAX 编程技术与实例. 北京：人民邮电出版社，2007.
11. 刘乃丽. 精通 Java EE 项目开发——基于 Eclipse、Spring、Struts、Hibernate. 北京：人民邮电出版社，2008.
12. 彭晖，史忠植. 语义 Web：让计算机读懂互联网. 计算机世界报，第 45 期，2007.11.26.
13. 明日科技. JavaScript 网页特效范例宝典. 北京：人民邮电出版社，2007.
14. 李刚. 基于 J2EE 的 AJAX 宝典. 北京：电子工业出版社，2007.
15. Java 开发技术十年的回顾与展望. 编程中国，http://www.bccn.net/Article/kfyy/java/.
16. 郝兴伟. Web 技术导论. 第 2 版. 北京：清华大学出版社，2009.
17. 甲骨文公司网站. http://www.oracle.com/us/sun/index.htm.
18. Apache 软件基金会. http://www.apache.org/.
19. 中国 XML 论坛. http://www.xml.org.cn/index.asp.
20. Java 中文世界论坛. http://bbs.chinajavaworld.com/index.jspa.
21. SOA 中国技术论坛. http://soachinaforum.com/.